

NPS ARCHIVE
1960
SMITH, B.

A SIMPLIFIED METHOD FOR DIGITAL
COMPUTATION OF CLOSED LOOP
FREQUENCY RESPONSE

BYERS G. SMITH
and
DONALD R. SCHAFER

DLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943-5002

A SIMPLIFIED METHOD
FOR DIGITAL COMPUTATION
OF CLOSED LOOP FREQUENCY RESPONSE

by

Byers G. Smith

B.S.E.E., Purdue University, 1944

B.S.E.E., U.S. Naval Postgraduate School, 1959

and

Donald R. Schaffer

B.S.E.E., U.S. Naval Postgraduate School, 1959

Submitted in Partial Fulfillment
of the Requirements for the
Degree of Master of Science
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
1960

A SIMPLIFIED METHOD
FOR DIGITAL COMPUTATION
OF CLOSED LOOP FREQUENCY RESPONSE

by

Byers G. Smith
Donald R. Schaffer

Submitted to the Department of Aeronautics and Astronautics
on May 21, 1960, in partial fulfillment of the requirements for the
degree of Master of Science.

ABSTRACT

This thesis develops a means by which a closed loop frequency response may be calculated by digital computer, given only the circuit configuration and component transfer functions. Using these data, a program is developed to perform the necessary logic for circuit analysis and obtain a performance function and frequency response over any desired frequency range, for an input and output at any specified circuit location.

The method is demonstrated by obtaining the performance function and frequency response of a complicated closed loop system.

Thesis Supervisor: Wallace Vander Velde

Title: Assistant Professor of
Aeronautics and Astronautics

ACKNOWLEDGEMENTS

The authors wish to express their appreciation to the personnel of the Instrumentation Laboratory, Massachusetts Institute of Technology, who assisted in this study, and in particular to Mr. Daniel Goldenberg for the original suggestion of and guidance in the execution of this paper; Mr. Charles Werner, Mr. Richard Russell and Mr. Robert Goodman and others of the staff of the Mathematics Group, for technical assistance, Professor Wallace Vander Velde of the Department of Aeronautics and Astronautics, for editorial comments and criticism and Miss Lillian Brouillette, for typing and proofreading the manuscript.

The graduate work for which this thesis is a partial requirement was performed while the authors were assigned by the United States Naval Postgraduate School for graduate training at the Massachusetts Institute of Technology.

TABLE OF CONTENTS

Chapter	1	Introduction	1
	2	General Description of System	3
	3	Detailed Description of Program Written for the IBM 650	22
	4	Summary of Procedures for Using the Program	50
Appendices			
	A	Program Details for Writing of XX-Array . .	60
	B	MAC Program	65
	C	Print-out of Performance Function	130
	D	Print-out of Frequency Response	139

LIST OF ILLUSTRATIONS

2.1.	Flow Chart of Broad Aspect of the Program	8
2.2.	Calculation of the Performance Function	10
2.3.	Detailed Flow Chart of Loop Routine	12-13
2.4.	Detailed Flow Chart of Path Routine	15
2.5.	General Flow Chart for Obtaining Performance Function Terms.	16
2.6.	General Flow Chart for Compiling Performance Function.	17
2.7.	Flow Diagram of Component Magnitude and Phase Angle Routine	19-20
3.1.	Block Diagram of Sample System	24
3.2.	G-Array of Coefficients of Transfer Function	40
4.1.	Block Diagram of Example	54
4.2.	Input Data Array of Example	55
4.3.	Transfer Function Array of Example	56
4.4.	Data Cards of Example	57

OBJECT

The object of this work is to determine the feasibility and devise a method for numerical computation of closed loop frequency response and to apply the method to an illustrative problem.

CHAPTER 1

INTRODUCTION

In the design and analysis of feedback control systems it is often necessary or desirable to know the steady state sinusoidal frequency response of the system. Various methods for obtaining this frequency response are available to the engineer, the most often used, perhaps, being those based on the Bode Diagram and Nichols Chart. For complicated high order systems, these graphical techniques are apt to become tedious and time-consuming; even more disadvantageous is that the time must be spent by the engineer, or at best, an assistant with knowledge of the engineering principles involved. There is also to be considered the human factor and its accompanying probability of error.

It would be advantageous then to have available a method requiring a minimum application by the engineer, one that is relatively fast, free from the human factor and, after the minimum initial application, could be handled by less-technically-qualified personnel. With the advent of high speed, large data capacity computers, it is reasonable to assume that such a method could be made available. It is toward this end that the method to be presented in this paper is directed.

In general, the minimum input information required from the design engineer is considered to be:

1. the transfer functions of the individual components of the system
2. a means of describing how the components are connected
3. the frequency range and desired increment of frequency over this range for which the response is required.

Item 1 is assumed known to the engineer and need only be entered in the form described in Chapter 2. Item 2 is desired to be as simple as

possible but still be of such a nature that a positive check may be made by the engineer to ensure that the information is entered correctly. This is described in detail in Chapter 2. Item 3 consists only of a statement of the values noted.

It is noted here that no new mathematical concepts are set forth, in fact, the simplest of arithmetic operations are involved. These basic fundamentals are the most readily adapted to digital computation. Graphical solutions such as the Bode Diagram were designed not because the fundamental mathematics were unknown, but because these simple arithmetic operations were so laborious as to be prohibitive. The modern computer performs these operations at an extremely rapid rate and permits programming the heretofore prohibitive methods.

Chapter 2 contains the physical interpretation, with appropriate block diagrams and flow charts, of how the program works; i.e., how the information is read into the computer and the general steps necessary to compute the frequency response. The physical reasoning and diagrams are intended to be general in that they describe the necessary programming steps and hence, are applicable to any digital computer. At the same time, they are in great enough detail to permit the program to be adapted to any computer with a minimum amount of additional labor.

Chapter 3 and appendices thereto contains the detailed program as written for the IBM 650 Computer and its associated equipment RAMAC. The "MAC" language in use at the Instrumentation Laboratory at MIT is utilized because of the relative ease of programming.

Chapter 4 contains a brief summary of the salient features of the program and its application to some sample systems.

The primary objective of this thesis is to set forth a general program, stressing simplicity of use and not necessarily to derive the optimum program with respect to minimum time and compactness. For this reason, it is realized that the program of Chapter 3 may not be the ultimate in computer programming but is general enough to be adapted to any modern digital computer.

CHAPTER 2

GENERAL DESCRIPTION OF THE SYSTEM

2.1. General

The program that is used to determine a general form of the system performance function and from this compute the frequency response of a linear, time invariant system is described in this chapter. The description is meant to be quite general in that it applies to programming any digital computer. If not general, the program would be of little value for any computer except the specific one used here. Even though general, a certain amount of detail is necessary to enable a prospective user to program the system without an unduly great amount of effort. The description in this chapter is taken by steps, beginning with a broad over-all aspect and block diagram flow chart and working down to the details. This gives the reader an opportunity to decide just how far he wants to go and permits comprehension at any desired level of detail.

2.2. Overall Description

As stated in Chapter 1, there are numerous methods in the literature and in current usage to calculate the frequency response. Some of these, particularly the graphical techniques, are not readily adaptable to digital computation. We may note, however, that in obtaining the system performance function by any method, we get at some stage of the calculation an equation of the form:

$$\frac{G_a G_b G_c + G_d G_e G_f + \dots}{1 + G_1 G_2 G_3 + G_2 G_3 G_4 + \dots}$$

where the G 's represent the transfer functions of the system components. A more careful study reveals that the components of the denominator terms are those components making up a loop or combinations of loops in the physical system. Although not quite so obvious, it turns out that some of the terms of the numerator may be recognized as loops and some as direct paths from the input to the output of the system. Therefore, it would seem that if we could determine the general scheme of how these loops and paths combine in the equation, then it might be possible to readily adapt this to digital computation. Mason (Ref. 5) obtains the general format of how they combine and arrives at a general equation representing the performance function for an input at i and an output at j as:

$$\frac{\theta_j}{\theta_i}(S) = \frac{\sum_k P_k \left[1 + \sum_r L_{rk} + \sum_r \Pi_2 L_{rk} + \sum_r \Pi_3 L_{rk} + \dots \right]}{1 + \sum_r L_r + \sum_r \Pi_2 L_r + \sum_r \Pi_3 L_r + \sum_r \Pi_4 L_r + \dots}$$

where:

P_k is the product of transfer functions of all components in a direct path from i to j . This product is defined as the "Path function" of path " k ".

L_r is the product of transfer functions of all components of a loop and is defined as the "loop function" of loop " r ".

$\Pi_2 L_r$ is products taken 2, 3, . . . at a time of loop functions of all loops not "touching" one another.

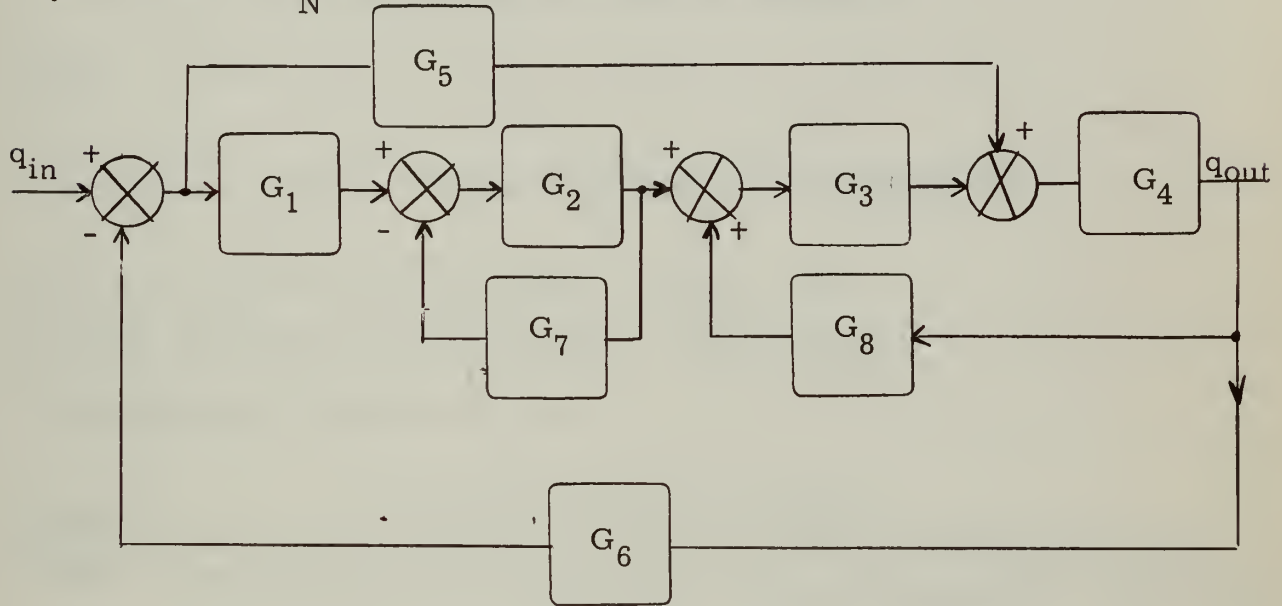
L_{rk} is the loop function of a loop not "touching" path " k "

$\Pi_2 L_{rk}$ is the products taken 2, 3, . . . at a time of the loop functions of all loops neither "touching" one another nor touching path " k "

The only rule involved is that of signs. This may be stated as: "In traversing a loop, if the number of (-) signs encountered is odd, the sign of the term is (+). If the number is even (including zero), then the sign of the term is (-)". This is readily understood from basic feedback system theory where we are familiar with the result that a negative feedback

function leads to a characteristic equation $1 + \text{the loop function}$. In traversing a "direct path" if the number of (-) signs encountered is even (including zero) the sign of the term is (+); if odd, the sign of the term is (-).

To demonstrate the use of the basic equation and the "touching vs. non-touching" character of loops and paths consider the following system where G_N is the transfer function of block (N)



From the block diagram, it is seen that there are:

- 1) 4 loops with loop functions

$$L_1 = G_1 G_2 G_3 G_4 G_6$$

$$L_2 = G_5 G_4 G_6$$

$$L_3 = G_2 G_7$$

$$L_4 = - G_3 G_4 G_8$$

$$\sum_r L_r = G_1 G_2 G_3 G_4 G_6 + G_5 G_4 G_6 + G_2 G_7 - G_3 G_4 G_8$$

- 2) 2 direct paths with path functions

$$P_1 = G_1 G_2 G_3 G_4$$

$$P_2 = G_5 G_4$$

The "touching" of 2 loops can also be seen by inspection but if in doubt the question might be asked "can a block be inserted in one of the loops without changing the loop function of the other?" If so, then they do not touch. In the block diagram above:

loop (1) touches the remaining three

loop (2) does not touch loop (3) but does touch (1) and (4)

loop (3) and (4) do not touch one another

then:

$$\sum_r \Pi_2 L_r = L_2 L_3 + L_3 L_4 = G_5 G_4 G_6 G_2 G_7 - G_2 G_7 G_3 G_4 G_8$$

Path (1) is seen to touch all loops: $\sum L_{r1} = 0$

Path (2) touches all except loop (3): $\sum L_{r2} = G_2 G_7$

The performance function then is:

$$\frac{q_{out}(S)}{q_{in}(S)} = \frac{G_1 G_2 G_3 G_4 + G_5 G_4 + G_5 G_4 G_2 G_7}{1 + G_1 G_2 G_3 G_4 G_6 + G_5 G_4 G_6 + G_2 G_7 - G_3 G_4 G_8 + G_5 G_4 G_6 G_2 G_7 - G_2 G_7 G_3 G_4 G_8}$$

To obtain the frequency response, two courses are now available. The component transfer functions might now be entered and a program written to expand the performance function above into the form

$$\frac{A S^n + B S^{n-1} + C S^{n-2} + \dots}{a S^m + b S^{m-1} + c S^{m-2} + \dots}$$

Then let $S = j \omega$ and compute the desired response. This method would require extensive working and storage space and might well exceed the capacity of some computers. The other course available is to enter

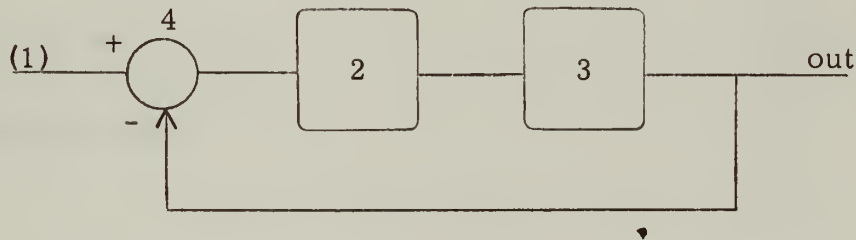


the transfer functions of the components, let $S = j \omega$ and compute a magnitude ratio and phase angle for each component for a particular value of ω . Then, substitute these values into the $\frac{q_{out}}{q_{in}}$ equation, perform the necessary mathematical operations and obtain the response for the particular value of ω . The program described in Chapter 3, uses the latter method.

The form in which the final answer is received will vary with the computer; perhaps punched cards, or if a plotter is available, then the final result may be read directly from a plot. A flow chart of this broad description is contained in Fig. 2.1.

2.3. Input Data

It is desired that this be in as simple a form as possible and provide some sort of a check to the programmer that the information (data) has been correctly entered. To fulfill both of these requirements, it is considered best to use an arbitrary numbering system of all components and junction points (summers). Then the inputs to a numbered unit are the numbers of the unit (s) from which they came. All that is necessary to be entered then is the number of the unit and the number (s) of the input. To permit operation by the computer, these data are entered in an array. For example, consider the simple system illustrated here in block diagram form:



If computer space is critical or if the system is extremely large, the blocks 2 and 3 may be just as well combined into a single unit. If considered as two separate units, the input array would be:

Unit	Inputs
2	4
3	2
4	1 -3 (the 1 is unnecessary but used here to illustrate the form for multiple inputs).

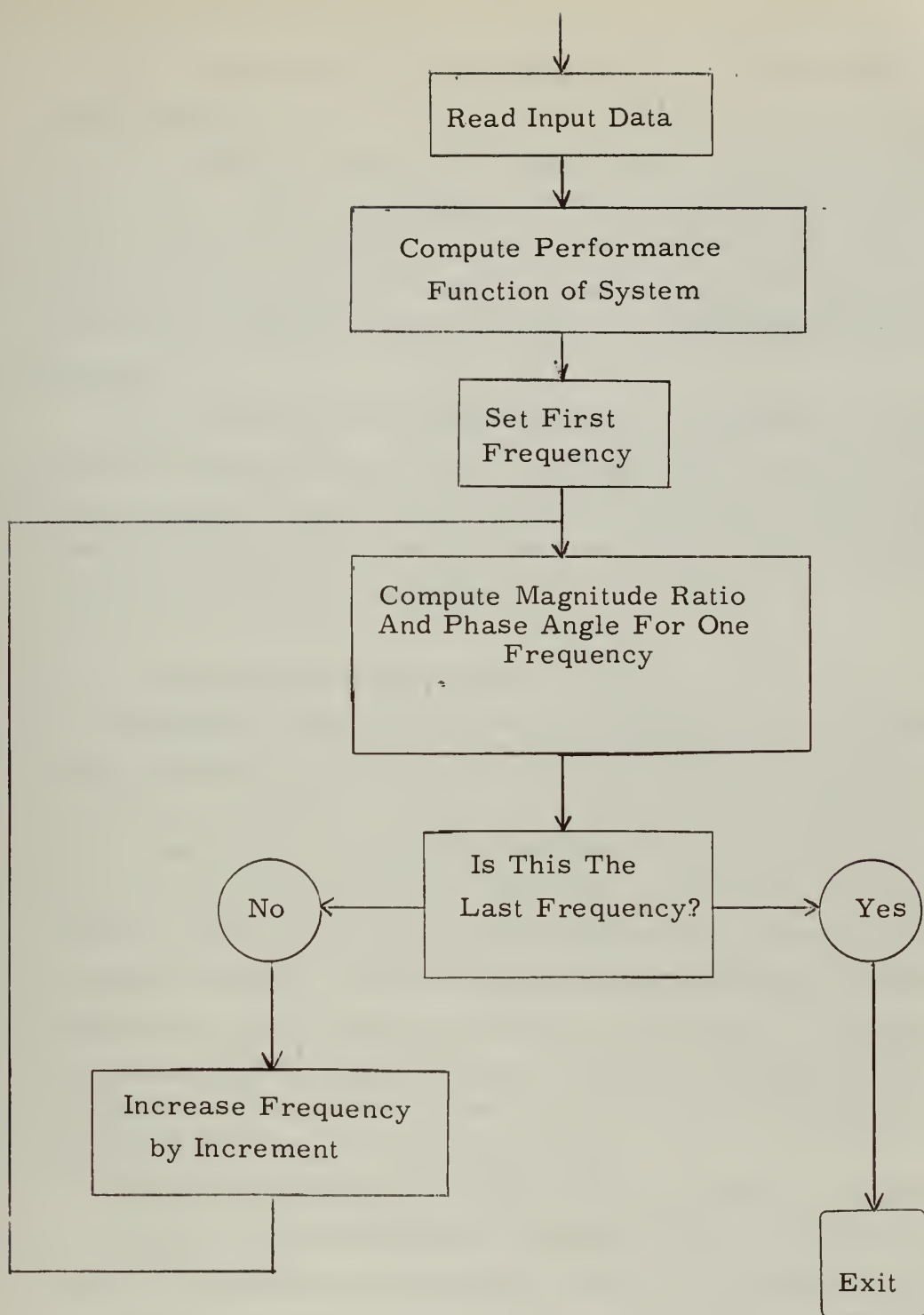


Fig. 2.1. Flow Chart of Broad Aspect of the Program

Note the (-) indicating this is to be subtracted in the summer; i.e. negative feedback.

The only other input data required are: input and output location numbers, in this case 4, 3; number of components and summers and maximum number of inputs to any unit. The reason for the first requirement is obvious; the last three are used to reduce computation time and conserve computer storage space by establishing the exact size of the array.

For reasons to be explained later, it is necessary to number the summers also and is convenient to assign larger numbers to these than the highest numbered component and list them last in the array. The particular order in which either components or summers are numbered is otherwise arbitrary.

2.4. Computation of Performance Function

Using the method described in paragraph 2.2., it is seen that what is actually required is a method of tracing through the numerical input data to determine the existent loops and direct paths of the system. These are found and recorded as groups of numbers; hence to determine the "non-touching" character, it is necessary only to check numbers of each group against the other groups. Since it is possible for paths or loops to "touch" without having common components, the necessity for numbering the summers is now seen. It is not possible for loops to touch without having at least one common junction point (summer) and vice versa. The broad scheme of this operation is described in the flow chart of Fig. 2.2.

After determining the existent loops, paths, non-touching loops, etc., the groups of numbers are compiled to form the general expression for the performance function. Since the requirement might exist for knowing the performance function, a subroutine is included in the main program for compiling and punching out this function, before commencing calculation of frequency response.

The methods used to determine the paths and loops are perhaps the heart of the entire program. The two methods are similar in general hence only the loop routine will be explained in detail and the

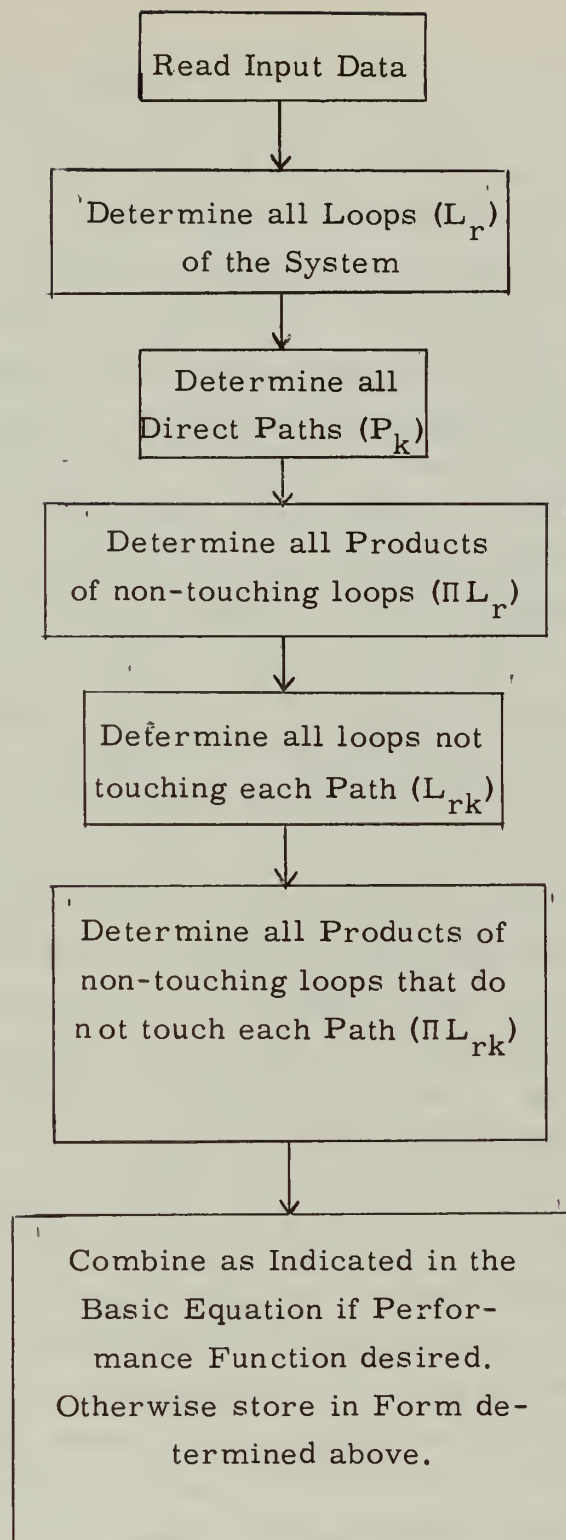
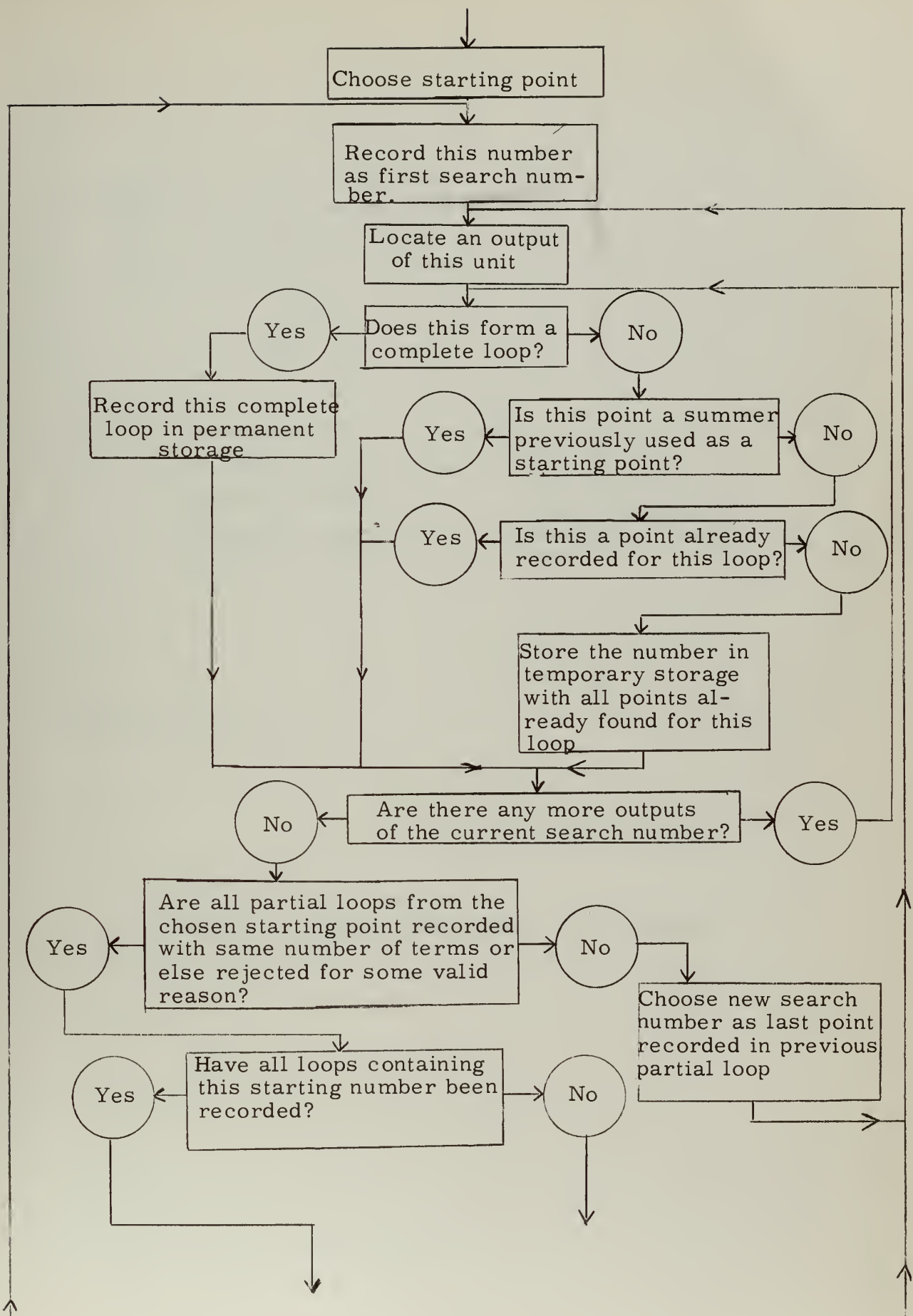


Fig. 2.2. Calculation Of The Performance Function

path routine in general except where it differs. The first step of the loop routine is to choose a starting point in the array. Since any loop of the system must contain a junction point (summer), then we choose as the starting point the first summer listed in the input data array (not necessarily the first one in the physical system). We could start with a component but can save time by considering only the summers. Hence, it is convenient to list the components first then the summers. This order might well be reversed but the program of Chapter 3 is written for the order noted. The first summer is located by knowing the number of components other than summers. To enable computation of the frequency response or in punching the performance function, the summer numbers are not desired. This is obviated by assigning higher numbers to the summers than to the components, and discarding all numbers greater than that of the highest numbered component. The computer, after noting the starting number, then scans the input array by rows until this number is found as an input to another numbered unit. Then the new number is used as the "search number" until it is found as the input to another numbered unit, and so on until we have a complete loop. It is seen now that difficulty might be encountered due to the output of a unit going to more than one other unit. This must be considered in the program. The way this is done is to record the first output then continue the search to see if there are any more. If more are found they too are recorded at this time. It is impractical to find one complete loop then go back to the starting point and search for more loops. Hence, all points in all branches are recorded at this time. This is more clearly described by the flow chart of Fig. 2.3. After recording (in the computer) all the loops containing the first summer, the program continues to the second and repeats the search and record process, ensuring that loops already recorded are not repeated, and so on until all loops are found. It is realized, of course, that checks must be made for such things as traversing a loop only once and recording only complete loops. This is also brought out more clearly in Fig. 2.3.



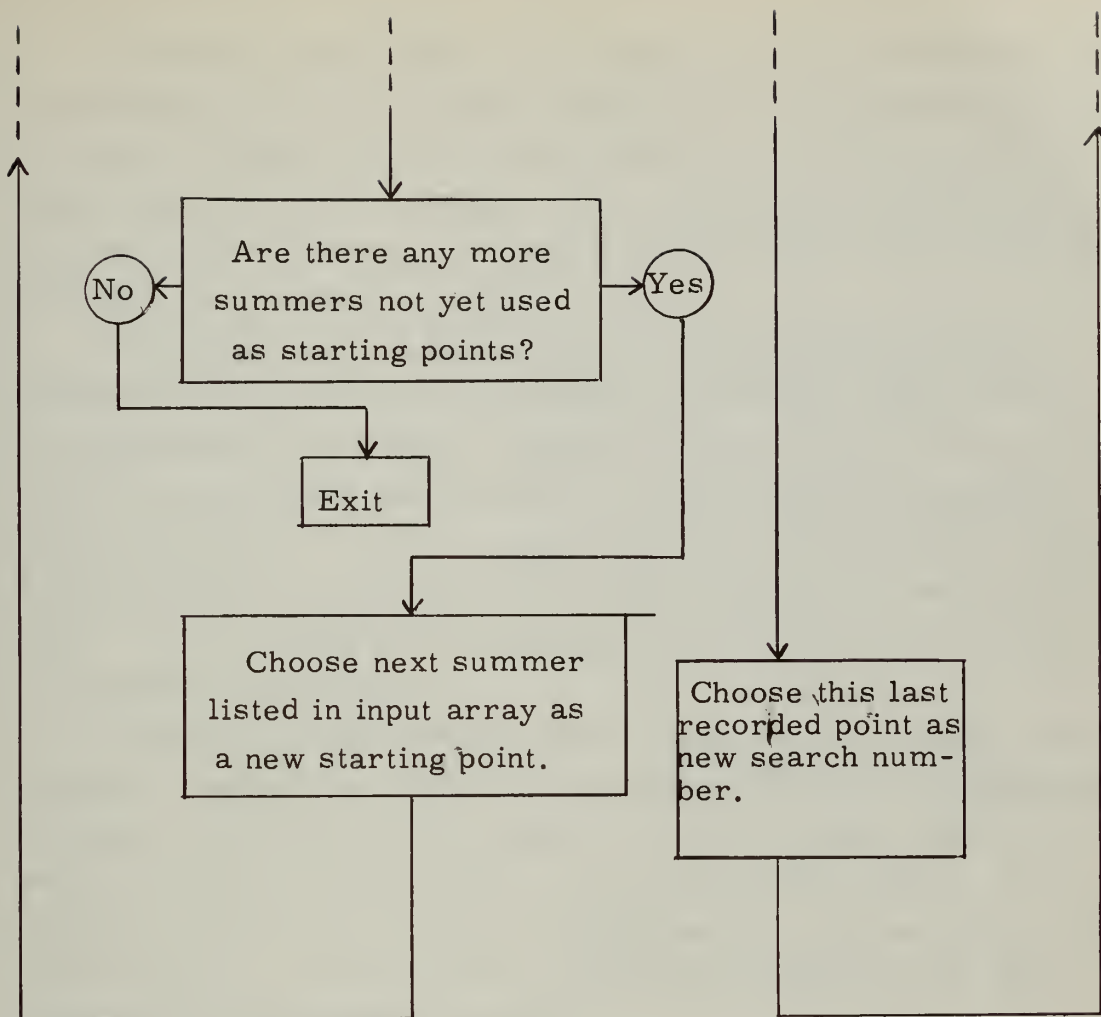


Fig. 2.3. Detailed Flow Chart of Loop Routine

The path routine differs from the loop routine primarily in that the starting point is the number inserted in the input data as the system input for which the response is desired, and also the end of the path is the listed output number. The similarities and differences are seen more clearly by comparing the flow chart for the path routine (Fig. 2.4.) with Fig. 2.3.

The importance of these two subroutines and the program sophistication necessary to accomplish the desired result cannot be emphasized too strongly. The flow charts of Figs. 2.3. and 2.4. clarify the procedure satisfactorily for understanding the general method employed. However, they are not considered to be in great enough detail to permit writing a similar program for another computer. For this reason, the method of this subroutine is explained in great detail in Chapter 3.

Having recorded all the existent loops and direct paths of the system, it remains to determine the remaining factors of the general equation. The idea is very simple; we merely check numbers of groups already recorded against appropriate other groups. If no identical numbers are found then the loops, etc., do not touch. Since this procedure is fairly readily understood, a general flow chart (Fig. 2.5.) is considered adequate.

To obtain the performance function, we now combine the appropriate previously stored numbers and punch out the function term by term. Fig. 2.6., is a general flow chart of this routine. It should be pointed out that since it is foreseeable that the performance function might not actually be desired, then writing this subroutine is optional. For this reason a "by-pass" is put into the program and effected by entering a value of zero for a variable read in with the input data. If the performance function is desired, then any number other than zero is entered.

2.5. Component Transfer Functions and Frequency Response

Any component whose input-output performance can be described by a linear ordinary differential equation with real constant coefficients has a transfer function which is rational in the Laplace variable S with the numerator and denominator polynomials expressible as

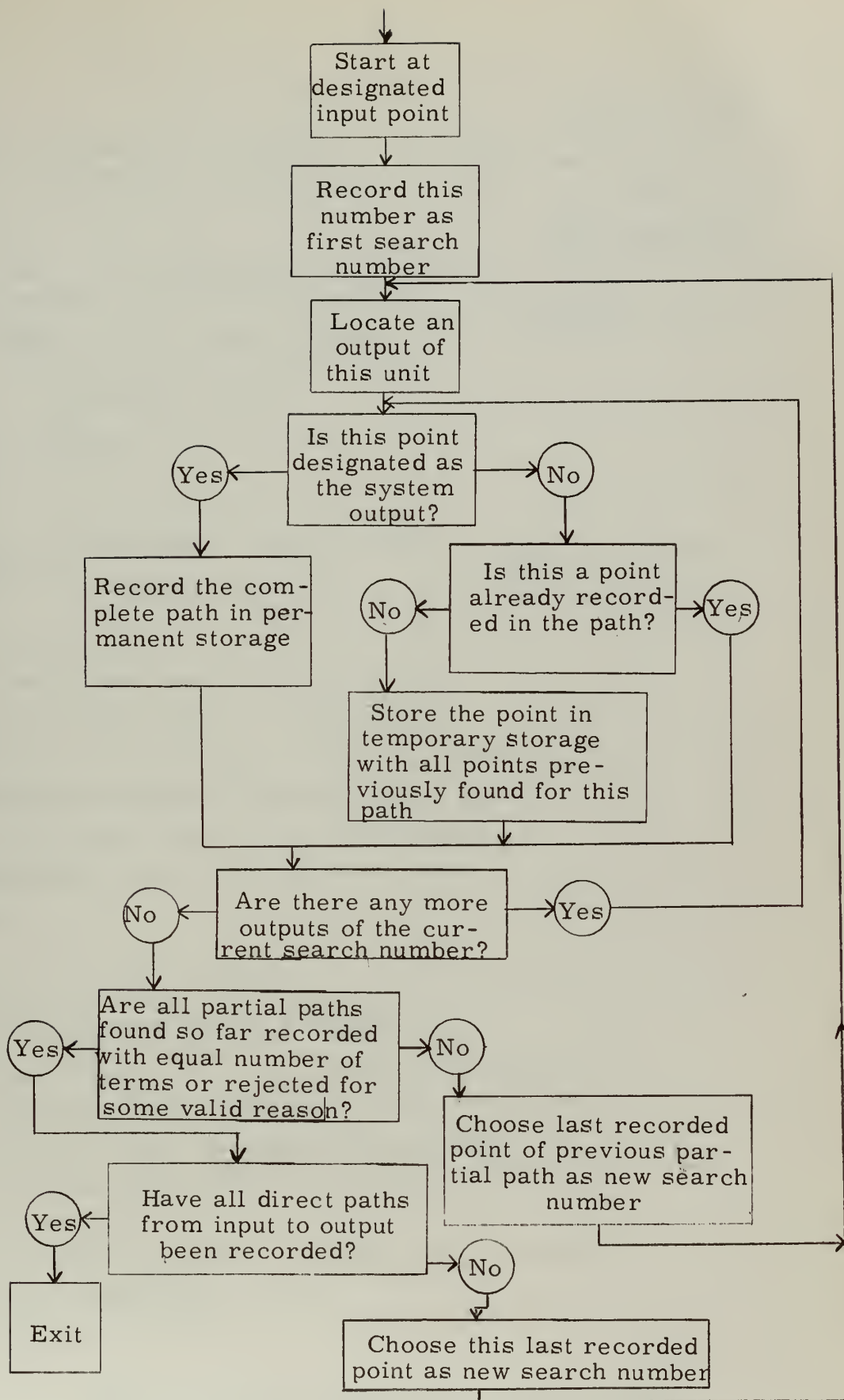


Fig. 2.4. Detailed Flow Chart of Path Routine

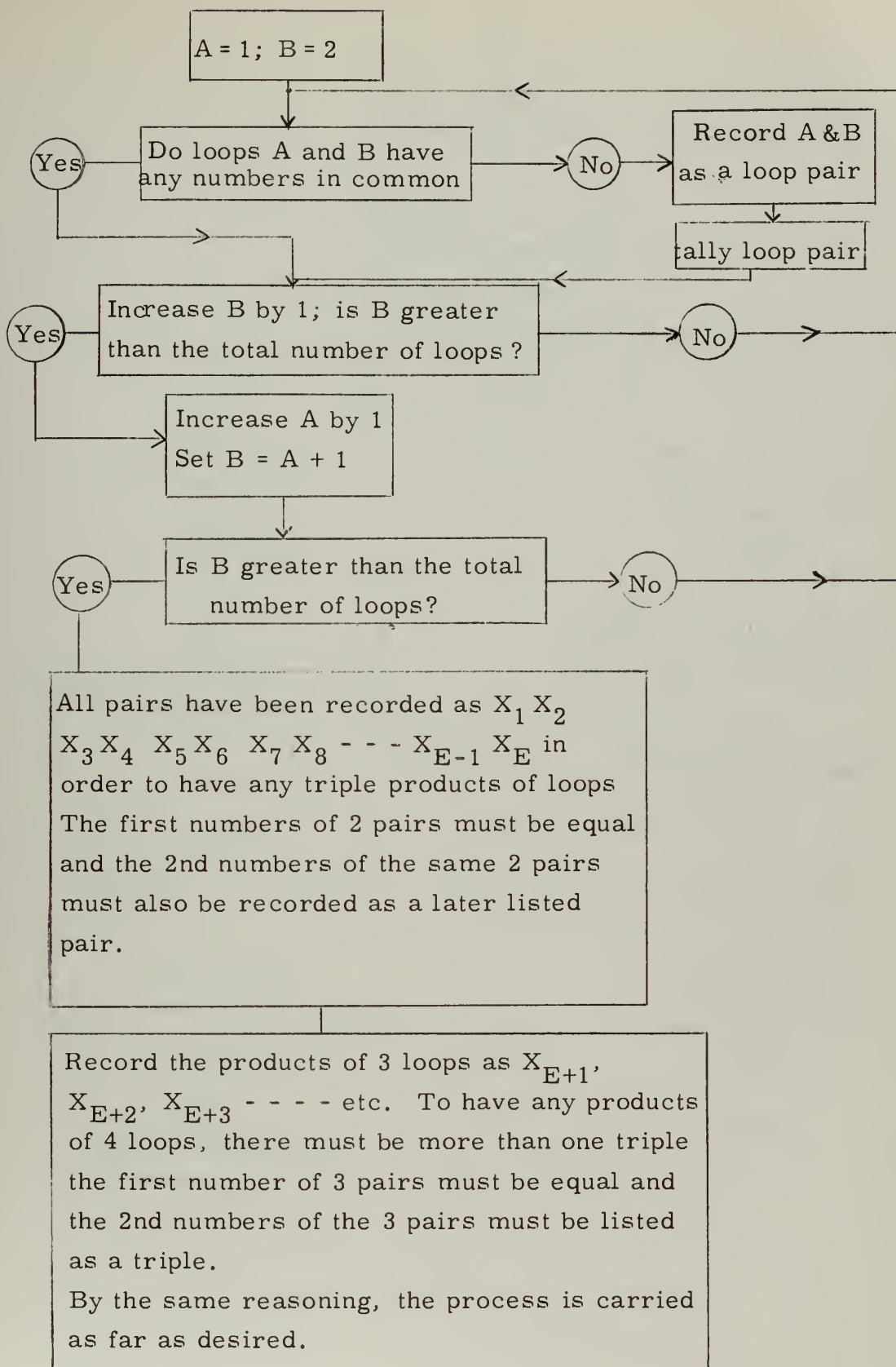


Fig. 2.5. General Flow Chart For Obtaining
The Performance Function Terms

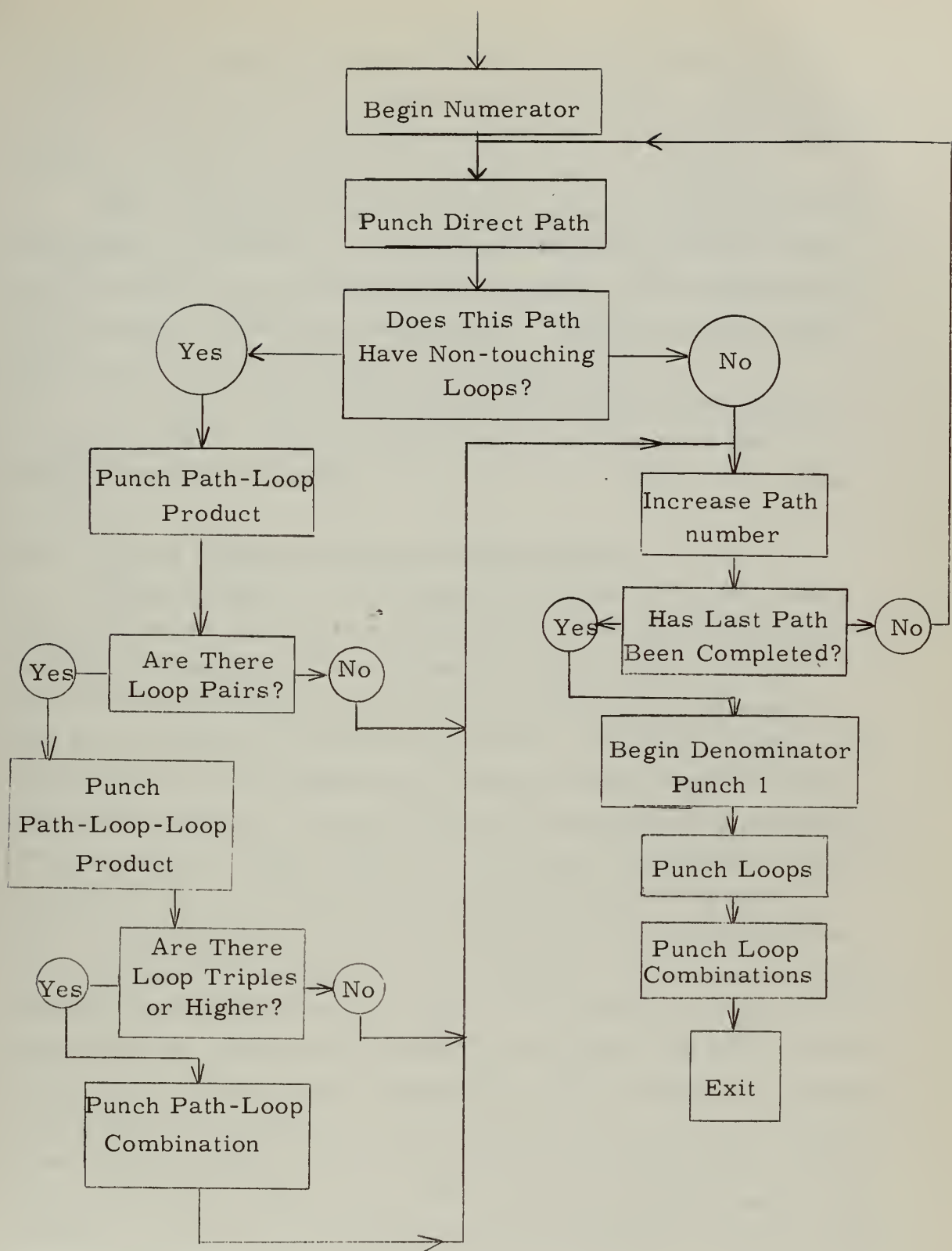


Fig. 2.6. General Flow Chart For Compiling
The Performance Function

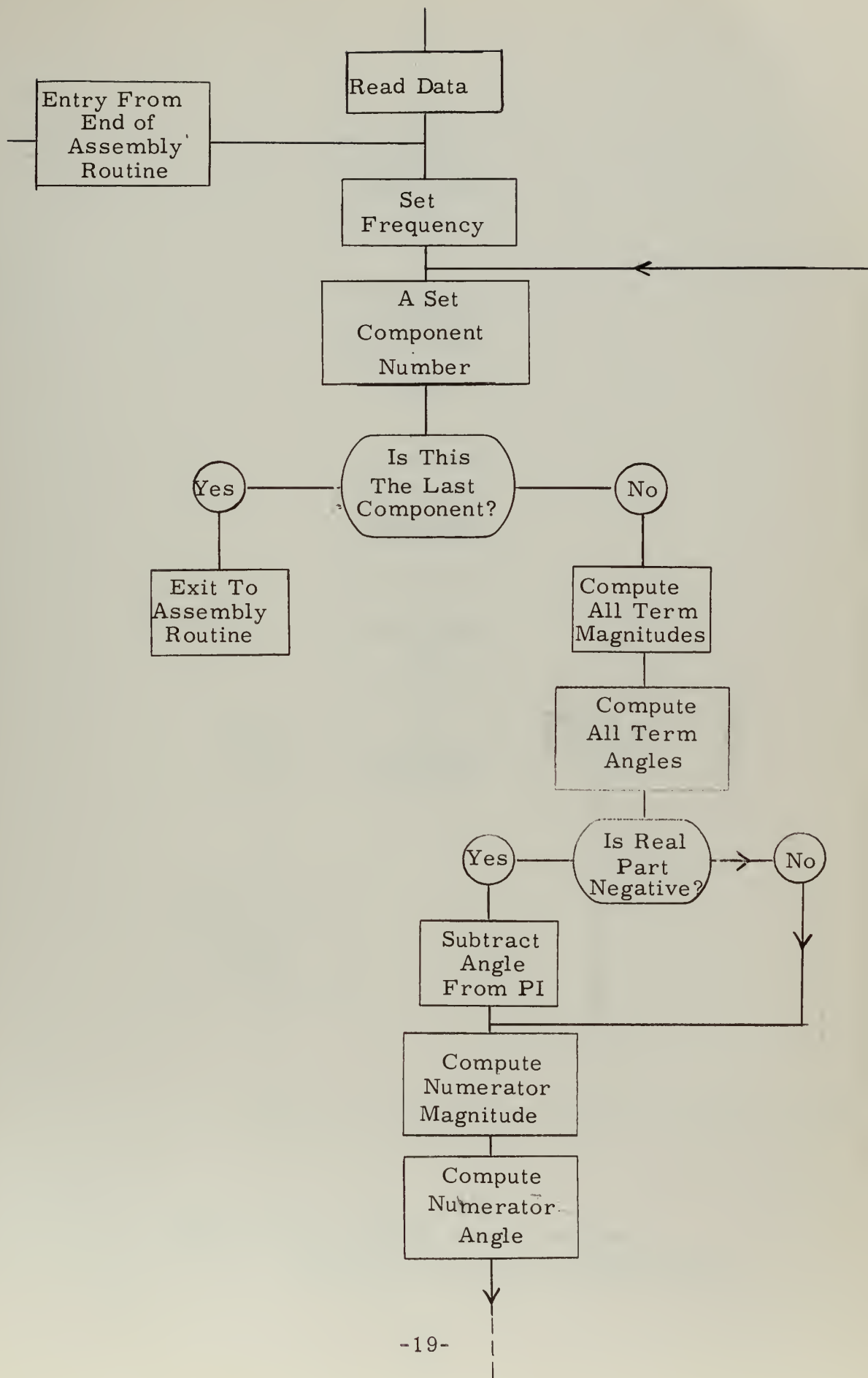
products of terms of the form $A S^2 + B S + C$ in which any of the coefficients may be zero. These transfer functions must now be entered. The form in which they are entered is discussed in detail with the program of Chapter 3 and illustrated in Chapter 4.

Next a magnitude ratio and phase angle is computed for this component by letting $S = j \omega$ and substituting the particular value of ω for which we are computing the response. The required bits of information regarding frequency are: frequency range (upper and lower limit) and increment. (See Fig. 2.7.).

To facilitate ease of recording the required information, a form is included with the program details of Chapter 3 and the sample system of Chapter 4.

2.6. System Magnitude Ratio and Phase Angle

Having computed, for the particular value of ω , the magnitude and phase angle of each component the next and final step is to substitute these values into the general performance function. If the entire performance function has been retained in storage, if and when punched, it could well be used to compute the system response; but, since it is desired to have this performance function calculation optional, then the routine for finding system response is written using the loops, paths etc., previously computed. This is a straight-forward routine easily written and understood once it is known where the information is stored from previous subroutines. Since the general equation from paragraph 2.2., contains only loops and paths and combinations of these, it is expedient to compute and store first the magnitude and phase for each loop and path, then use these stored values in the combinations. The performance function is in general a summation of loops, paths, etc., hence a conversion from polar to rectangular form is also required and, after compiling, a conversion back to polar form. For any greater detail, the reader is referred to the program detail of Chapter 3 and the sample problem of Chapter 4.





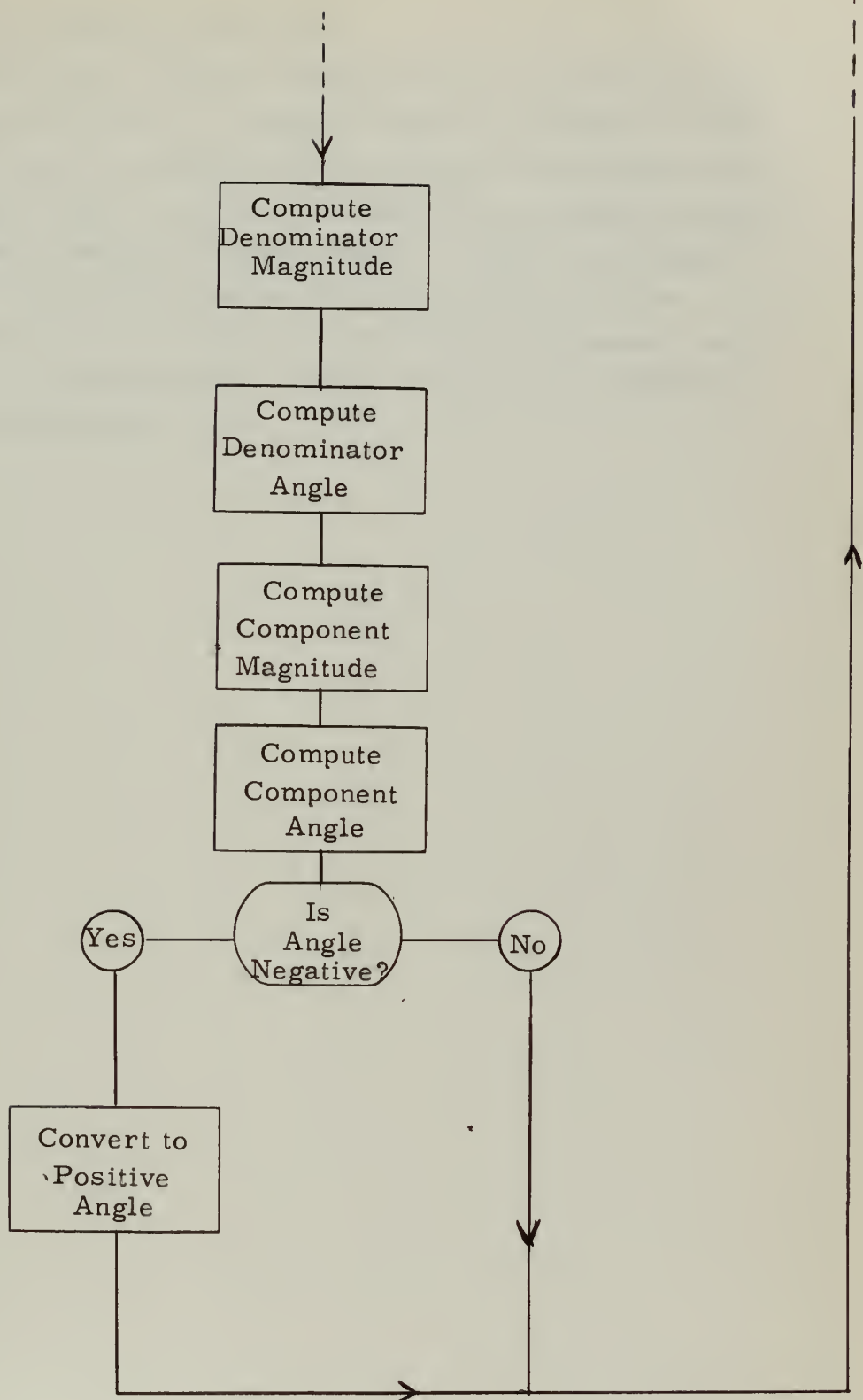


Fig. 2.7. Flow Diagram of Component Magnitude and Phase Angle Routine



2.7. Frequency Response "Re-Run"

After obtaining a frequency response over an indicated range with a specified increment, it might be desired to change the range, increment or even the transfer function of some of the existent components. In this case, it would not be desired to recompute the general performance equation of paragraph 2.2., hence, provisions are made to re-enter the program with these different values. The method will vary with the computer used and for this reason, is not detailed here. A detailed step by step procedure for the IBM 650 is contained in paragraph 3.10.



CHAPTER 3

DETAILED DESCRIPTION OF PROGRAM WRITTEN FOR IBM-650

3.1. General

It was stated in Chapter 1, that the MAC language (Ref. 6) is used in this program for simplicity of construction. Very briefly, MAC may be described as follows: A plug board wiring scheme permits recognition of certain special plain language commands such as "PUNCH", "GO TO", "IF", "DO" and "DO TO". Arithmetic symbols are also recognized; examples of these are the signs for "equals", "plus", "minus", "decimal point" and "parenthesis". Any number of parentheses may be used consistent with proper arithmetical usage.

Variables are created by several means, the most common one being an equation with the variable to the left of the equal sign, e.g., $A = 2.5 + B/2$ is interpreted as; "a new variable (or new value of a variable in use) is equal to 2.5 plus (B (a previously defined quantity) divided by 2). Upon encountering this command in a program, the value of A is computed and stored in the memory and may be used at any time in the right hand side of an equation.

Routines for certain other functions, such as trigonometric functions, square root, absolute value, integer are built into the basic storage and are referred to upon recognition of the functions: SIN, COS, TAN, ARCSIN, SQRT, ABS, INTEGER and others for which the reader is referred to Ref. 6 of the bibliography. Any number of subscripts, either numbers or letters for which a value has been computed, may also be used. It is noted that subscripted variables are treated in the same manner as non-subscripted ones and new variables may be introduced accordingly. If the subscript is itself a variable, its value must be defined. The subscript is then an "index" and must be

so indicated on the first card of the program. Variables are limited to letters of the alphabet, numbers, and combinations of either or both. The physical limits imposed by MAC is a total of 800 variables.

The individual lines and cards of the MAC program are numbered in sequence, using the prefixes E, M, S to indicate exponent, main, and subscript cards respectively. Equation numbers when required appear to the right of the MAC statements and equations, and are used as a means of transferring control to selected points in the program or to establish loops.

After all MAC program cards are read into the computer, an internal assembly routine converts this program into the basic language of the computer, begins at a designated point, and proceeds with the computation. This "new program" may be punched on cards as it is being written, the cards then being referred to as a "load deck"; or if no future need is anticipated, the load deck is not punched. Use of this load deck in lieu of MAC cards for future operations reduces computation time considerably.

3.2. Major Subroutines of the Program

Reference to the basic equation for determining the performance function, Paragraph 2.2., shows that the component numbers are required for:

- a. All closed loops of the system (L_r)
- b. All direct paths from input to output (P_k)
- c. All combinations of 2, 3, 4 etc. loops not touching one another (L_r products)
- d. All loops not touching each path (L_{rk})
- e. All combinations of 2, 3, 4 etc. loops not touching each other and not touching each path (L_{rk} products)

In addition to the above, other major subroutines are:

- f. Calculation and punching of the performance function
- g. Calculation of frequency response (for each ω) of:
 1. All components
 2. All loop functions, all path functions, certain products of loop and path functions
 3. Denominator of the system response
 4. Numerator of the system response
 5. The system

Flow charts for each of these have been included in the general discussion of Chapter 2. The complete program is appended as Appendix B and is discussed in detail in the following paragraphs. Each major subroutine will be explained separately, with reference to the appropriate flow chart and sections of the program. The numbers in parentheses following the subroutine paragraph title indicate the appropriate location in the program.

3.3. Input Data

Before discussing the program, it is essential that the "Input Data Array" be understood. This is most readily accomplished by means of a block diagram of a hypothetical system:

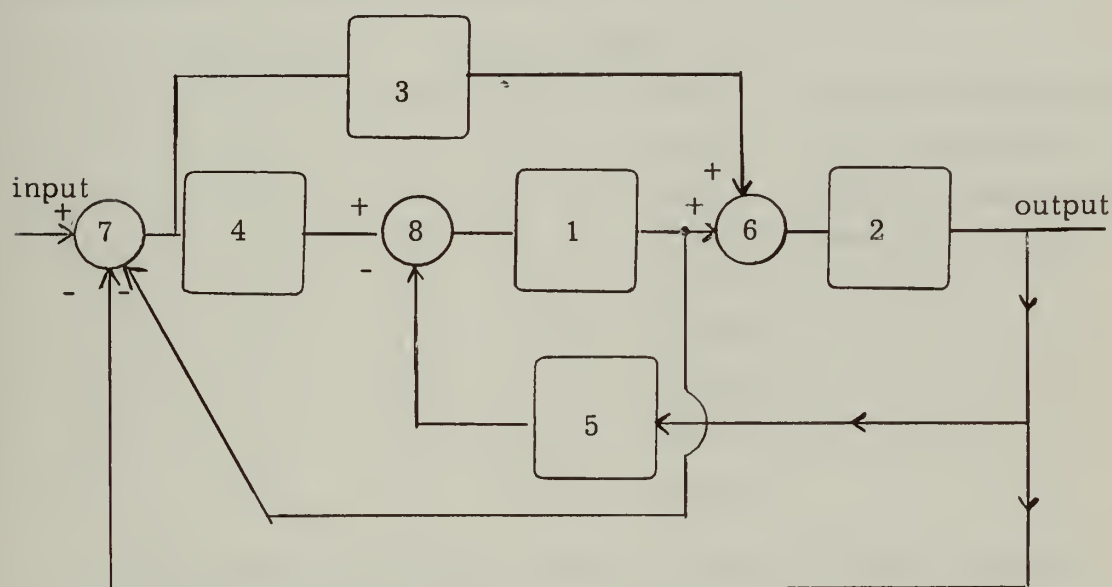


Fig. 3.1.

The numbering of components and summers is shown to be purely arbitrary except that summers are assigned higher numbers. Data Cards for MAC contain six or less numbers and are written in floating

point form (Ref. 7). The first card contains in the order listed:

Maximum no. of inputs to any unit (excluding the system input)	2
Number of component boxes	5
Number of summers	3
Input unit number	7
Output unit number	2
Is performance function desired	0 for No. 1 for Yes

On the succeeding cards, the inputs to each unit are listed, using a separate card for each unit. Again the order of listing is arbitrary except that the summers should follow the component boxes:

Unit No.	Inputs to this unit				
1	8	0	0	(0-3)	(numbers in parentheses
2	6	0	0	(4-7)	are not entered, but are
3	7	0	0	(8-11)	used for explanation fol-
4	7	0	0	(12-15)	lowing)
5	2	0	0	(16-19)	
6	1	3	0	(20-23)	
7	-2	-1	0	(24-27)	
8	4	-5	0	(28-31)	
0	0	0	0	(32-35)	

The zeros are required since with MAC a "Read" command is used to enter data and this Read command must specify what values and how many are to be read. The program for which this Chapter is written uses the "maximum number of inputs to any unit" as the basis for specifying the width of the array and hence, the number of values to be read. The zeros are used as "stops" in the search routine. Note, the (-) signs indicating negative feedback into unit 7 and 8.

For any system, a rectangular array is obtained, the width of which is the "maximum number of inputs to any unit" + 2. The length of the array is "the number of component boxes + the number of summers + 1 (the last row is all zeros).

Each position in the array is identified as a subscripted variable and the values read from the data cards are thereafter recognized as the value of the variable. The numbers in parenthesis are the subscripts identifying the locations. G is chosen as the variable to be subscripted. For example, the value of G_{22} is 3.

3.4. L_x Subroutine (M 0075 - M 0585)

It is desired to determine the units comprising each of the closed loops of the system. As noted in Chapter 2, this is a very important but complicated subroutine. To ensure complete understanding requires that the program be explained in great detail. Since this is, of necessity, a lengthy discussion the program details are included as Appendix A and only the method will be discussed here. Refer to the flow chart of Fig. 2.3. The basic method is to choose as a starting point in the array, the first summer listed. Using this number as a "search number" the input array is scanned, starting with G_1 until the number is found as an input to another unit. The search number and the number of the output unit are entered temporarily in an array designated by the subscripted variable XX. The search continues for more outputs of the "search number". If any more are found, a new line is formed in the XX-array for each. After all outputs are recorded, the last output is chosen as a new search number and the same procedure carried out again, writing a new line in the XX-array each time a new unit is found to be a point in the loop. Although not too obvious at this time, this is necessary to ensure the same path is not being traversed around a loop, and to have all numbers available when the loop is found to have been completed.

To illustrate briefly, the first summer listed in the array of paragraph 3.3., is (6) which goes to (2), so the XX-array is:

line (a)	6
line (b)	6 2

No more outputs of 6 are found so using (2) as a search number, its output is found to go to (5) and negatively to (7). These are recorded

in the XX array as lines (c) and (d):

line (c)	6	2	5
line (d)	-6	2	7

(7) is the next search number, for which an output is found to 3 and 4. These are recorded as lines (e) and (f). Then (5) is selected as a search number, the output of which feeds negatively to (8), (line g)

line (e)	-6	2	7	3
line (f)	-6	2	7	4
line (g)	-6	2	5	8

The routine continues, checking each new number to see if it forms a completed loop. If so, the appropriate line of the XX-array is recorded permanently as a loop, for which the subscripted variable ZZ is used. Continuing the search process yields one new line from (g) and one from (f) but when (3) from line (e) is checked, it is found to go to (6) which is the starting point, hence a complete loop has been found and a new line is not entered in the XX-array, but is recorded as ZZ_1 to ZZ_4 with values of -6, 2, 7, 3 respectively. ZZ_5 is then recorded as zero as a separator from the next loop.

After each number is located, it must be checked to see that it has not already been entered as a point in the line of the XX-array. If so, this represents a minor loop within the main loop that is being traced. This minor loop will be recorded as a line in a succeeding array. Hence, no new line is entered. This is illustrated as follows:

line (h)	-6	2	5	8	1	
line (i)	-6	2	7	4	8	
<hr/>						
line (j)	-6	2	7	4	8	1

Now use (1) from line (h) as a search number and find it goes to (6), a complete loop, and back negatively to (7). A new loop ZZ_6 to ZZ_{10}

is recorded as -6, 2, 5, 8, 1 and a new line (k) is recorded as:

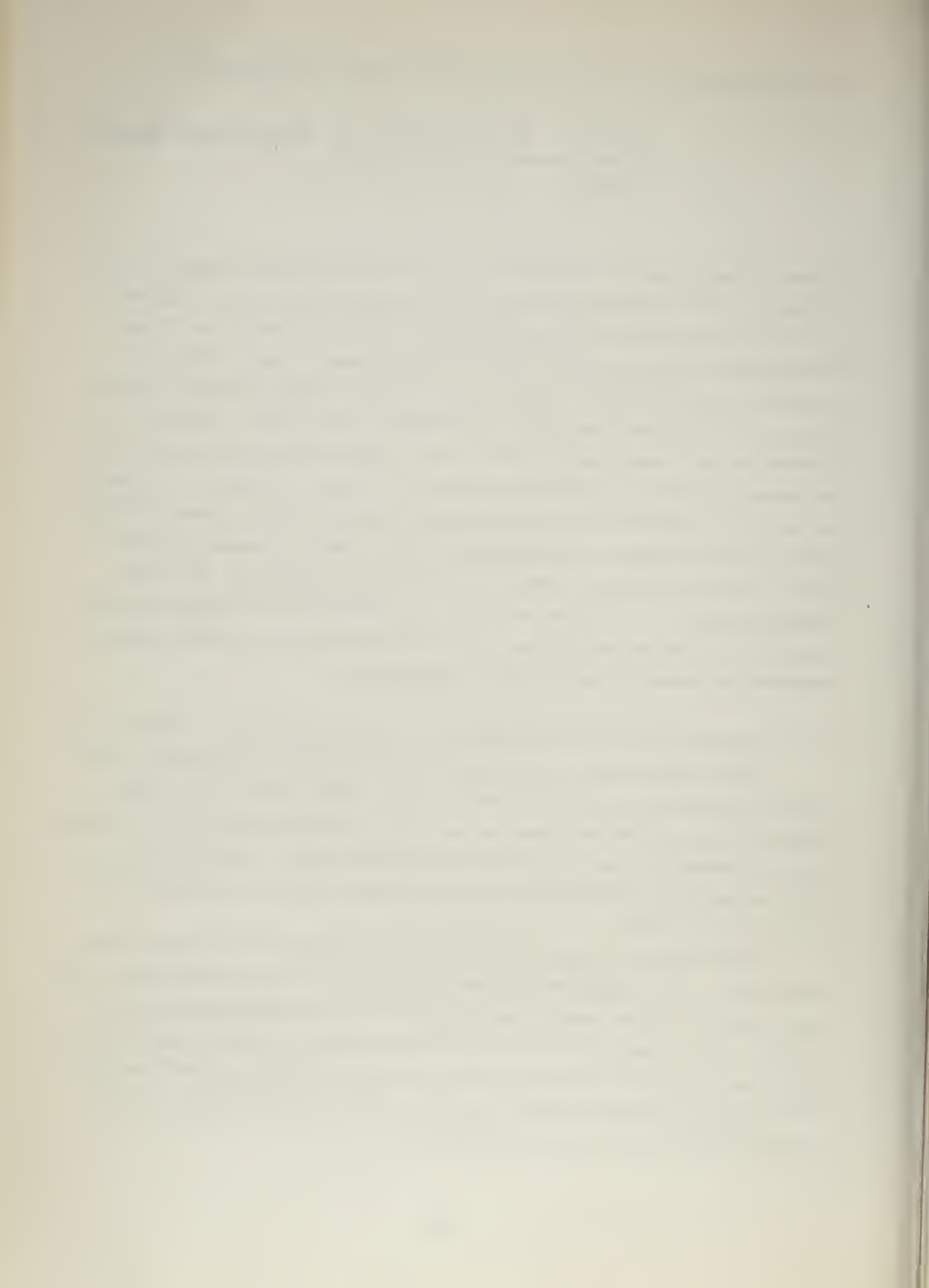
line (k)	+6	2	5	8	1	7	(note the sign change)
line (l)	6	2	5	8	1	7	3
line (m)	6	2	5	8	1	7	4

Lines (l) and (m) are recorded from line (k) and we move back to the (1) of line (j). (1) is found to go to (6), a complete loop, hence, is recorded in the ZZ list but not in the XX-array, and to (7); but since (7) has already been recorded in this line, it is disregarded and nothing is recorded in the XX-array. The process continues until all loops of which summer (6) is a part, have been recorded in the ZZ list. The next summer of the "input data array" is then chosen as a starting point and the same procedure is carried out as for the first summer. It is realized now that loops that have already been recorded from summer (6) must not be recorded again, so appropriate checks must be made to prevent this. When all summers have been checked for new loops, all of the existent loops of the system have been recorded and the program moves on to the next major subroutine. For greater detail as to how the above methods are programmed, refer to Appendix A.

3.5. Subroutine for Determining Direct Paths (PK routine; MO590-1040)

This routine differs only slightly from the LR routine above, and will be discussed in detail only where major differences occur. The search of the G-array is the same as above except the idea now is to find all components in a path between two specified units. Therefore, the first number of the XX-array is the designated input unit and the array is written only once.

Recording new lines in the XX-array (MO790-0820) and the checks made prior to this (MO735-0785) are the same as before except here, the check for a complete path is made against the specified output and the duplicate point check includes the starting number. When complete paths are found, all units of the path are recorded as the subscripted variable YY and this procedure differs from the LR routine only in that the last number found is also recorded (MO970).



It is necessary to know the number of paths just as it was necessary to know the number of loops, hence a tally is made each time a new path is recorded (M1020 and M1040). Note that Y_0 is the number of paths and the subscripted variable Y also is used in the XX-array. The reasoning here is that the variables used in writing the XX-array are not needed after this subroutine is completed and with limited computer space it is best to use variables that may also be used later. Throughout the program, the single letter variables are used many times but space is reserved for a designated number of variables with indexed subscripts (i.e., subscripts that are also variables) and, once used, this space is wasted unless the subscripted variables are chosen such that they may be used again. Since, as will be seen later, there are a number of constants associated with each path, (e.g. the number of loops not touching each path), it is convenient to use an indexed-subscripted variable to denote these constants. The subscripted variable Y is used for this and hence that space used by the Y 's in writing the XX-array is not wasted. The only possibility of an overlap is Y_0 . Therefore, the tally of paths is kept as a single letter variable (M1020) and upon completion Y_0 is set equal to this tally.

3.6. Products of Loop Functions Subroutine (M1050-1795)

From the basic performance function equation of Chapter 2, it is seen that both the numerator and denominator are made up of a series of terms, the number of which is determined by the complexity of the physical system. It is impractical to program an unlimited number of product terms and the question quite naturally arises as to how many should the program be capable of handling. Intuitively, it is seen that the degree of complexity of the physical system also determines the amount of computer space required for the "input data array", the XX-array, the ZZ and YY lists, etc. There is therefore, a physical limit imposed on any program written in MAC without resorting to other means for obtaining computer space. Consideration of these limiting factors leads to an estimation of the maximum number of product terms that should be programmed. The program appended to this report is written for a maximum of "products of loop functions taken four at a time of all loops not touching one another".

The method used to determine the touching vs. non-touching character of the loops is to check each number (component or summer) recorded in a loop against each number of every other loop. If no identical numbers are found in the two loops, then they do not "touch" and are to be so recorded. The details of the program are most readily explained by means of an example.

Consider a system containing four loops. These have been recorded in the ZZ list with a zero separating the numbers of one loop from the next:

(1)	(2)	(3)	(4)
ZZ ₁	ZZ _{a + 2}	ZZ _{b + 2}	ZZ _{c + 2}
	.	.	.
ZZ ₂	.	.	.
.	.	.	.
ZZ _a	ZZ _b	ZZ _c	ZZ _d
ZZ _{a + 1} = 0	ZZ _{b + 1} = 0	ZZ _{c + 1} = 0	ZZ _{d + 1} = 0

Check loop (1) against loop (2) by first checking ZZ₁ against ZZ_{a + 2} to ZZ_{b + 1}.

For convenience of locating loops by number at a later time, the first ZZ number of each loop is recorded as the check proceeds. The subscripted variable G is used for this since at the present time, the space reserved for the G's is unused. Since a + 2 is not yet known, it is found by checking the ZZ list until a zero is found (M1135-1140). The next ZZ number after the zero is ZZ_{a + 2} and G₂ is recorded as this value (M1160). Now the check consists of: ZZ₁ (first check point) against all numbers of loop 2. When ZZ_{b + 1} (= 0) is found, pick up ZZ₂ as the "check point" and start again with ZZ_{a + 2} as the "check number". The procedure is continued until ZZ_{a + 1} (= 0) is found as the check point or until the check number and check point are identical. Since algebraic signs are of no significance, absolute values are used in the check.

If the check point becomes zero, loops (1) and (2) are recorded as a pair of non-touching loops, and loop (1) is then checked against

loop (3) and so on. When the loop number of the loop to be checked exceeds Z_0 (number of loops) (M1150) the procedure is to check loop (2) against the remaining loops in the same manner as before, except now the first ZZ number of each loop is known and used as the first "check number". The procedure continues until loop number ($Z_0 - 1$) is checked against loop number (Z_0).

The non-touching loop pairs found in the above process are recorded as the subscripted variable XX, beginning with XX_0 . The number of "loop pairs" is recorded as Z_1 (M1365).

To determine the "loops not touching each other, taken 3 at a time" (hereafter referred to as "triple products" or "loop triples"), it is now easier to use the loop numbers of the "loop pairs". These pairs are recorded as XX_0 to XX_{2Z_1-1} .

The procedure is more readily explained by an example. Consider 5 loop pairs recorded as:

(1)	(2)	(3)	(4)	(5)
XX_0 XX_1	XX_2 XX_3	XX_4 XX_5	XX_6 XX_7	XX_8 XX_9
A C	B B+1	D D+1		

The variables A, B, C, D are defined initially as the XX subscripts as indicated. In order to have any "triple products" one number of each of two pairs must be the same. By the manner in which the pairs were recorded, the first number of the pair is always less than the second and is always equal to or less than the first number of each succeeding pair. Therefore, in order for loop number XX_A to be one of a loop triple, XX_B must equal XX_A . If not, then all four variables are increased by two and the check is made again. If $XX_A = XX_B$, in order to have a loop triple, loops XX_C and XX_{B+1} must not touch each other, and hence will be recorded somewhere farther on as a pair. Check for this by using XX_D . For this to occur $XX_D = XX_C$, if not then D is increased by two and checked again (until $D = 2Z_1$). If $XX_D = XX_C$, then to have a loop triple XX_{D+1} must equal XX_{B+1} . If

this is found to be true, the loop numbers of the three non-touching loops are XX_A , XX_C and XX_{B+1} . These three numbers are recorded as subscripted XX variables beginning with XX_{2Z_1} . The number of triple products is recorded as Z_2 (M1570).

The "loops not touching each other, taken four at a time" (hereafter referred to as "4's" or "products of four") are found by an extension of the above process.

Consider the same 5 loops pairs as above, and two triple products:

(1)	(2)	(3)	(4)	(5)
XX_0 XX_1	XX_2 XX_3	XX_4 XX_5	XX_6 XX_7	XX_8 XX_9
A C	B	D		

(1)	(2)
XX_{10} XX_{11} XX_{12}	XX_{13} XX_{14} XX_{15}
F	

For a "product of four" to exist $XX_A = XX_B = XX_D$ and XX_C XX_{B+1} XX_{D+1} do not touch each other. If the latter statement is true, then XX_C , XX_{B+1} , XX_{D+1} have been recorded as a triple product and the program follows a procedure similar to that described for the triple products. If any "4's" are found, the loop numbers are XX_A , XX_C , XX_{B+1} , XX_{D+1} , and these are recorded as XX variables immediately following the loop triples. The first loop number of the "4's" would be $XX_{2Z_1 + 3Z_2}$. The number of "4's" is recorded as Z_3 (M1795).

3.7. Loops and Products of Loops not Touching Paths (M1805-2500)

The same reasoning regarding the extent of loop products routines applies equally to this subroutine and the physical limitation is placed at having no more than products of three loops not touching one another and not touching a path. This is not considered a limiting factor since the complexity of the system with greater than this number of loop products not touching paths would be such that other limitations previously mentioned would prevail.

The method used here is similar to that of the L_r subroutine. The procedure is to find all loops not touching path 1, then all loop pairs and loop triples not touching path 1. After these are stored, the process starts again with path 2 and so on until completion.

Consider the following as having been previously recorded:

<u>2 direct paths;</u>		<u>4 loops</u>			
(1)	(2)	(1)	(2)	(3)	(4)
YY_1	YY_{A+2}	ZZ_1	ZZ_{C+1}	ZZ_{D+1}	ZZ_{E+1}

YY_2
.
.
.
YY_{A+1}	$YY_{B+1}=0$	$ZZ_C=0$	$ZZ_C=0$	$ZZ_E=0$	$ZZ_F=0$

<u>3 pairs</u>	<u>1 triple</u>	<u>no "4's"</u>
loops	loops	
1, 2	1, 2, 4	
1, 4		
2, 4		

The pairs and triples are stored, as noted before, as :

XX_0	XX_1	XX_2	XX_3	XX_4	XX_5	XX_6	XX_7	XX_8
1	2	1	4	2	4	1	2	4

and the following variables have been previously defined:

Y_0	number of paths	= 2	
Z_0	number of loops	= 4	
Z_1	number of pairs	= 3	:
Z_2	number of triples	= 1	"
Z_3	number of "4's"	= 0	

THE UNIVERSITY OF CHICAGO
DIVISION OF THE PHYSICAL SCIENCES
DEPARTMENT OF CHEMISTRY
530 SOUTH EAST ASIAN AVENUE
CHICAGO, ILLINOIS 60607
TEL: 773-936-5000
FAX: 773-936-5001
WWW: WWW.CHEM.UCHICAGO.EDU

1. Name of the donor: [Name]
2. Address: [Address]
3. City: [City] State: [State] Zip: [Zip]
4. Country: [Country]
5. Telephone: [Telephone]
6. Fax: [Fax]
7. E-mail: [E-mail]
8. Date of birth: [Date]
9. Date of death: [Date]
10. Date of donation: [Date]

11. Name of the recipient: [Name]
12. Address: [Address]
13. City: [City] State: [State] Zip: [Zip]
14. Country: [Country]
15. Telephone: [Telephone]
16. Fax: [Fax]
17. E-mail: [E-mail]
18. Date of birth: [Date]
19. Date of death: [Date]
20. Date of donation: [Date]

21. Name of the donor: [Name]
22. Address: [Address]
23. City: [City] State: [State] Zip: [Zip]
24. Country: [Country]
25. Telephone: [Telephone]
26. Fax: [Fax]
27. E-mail: [E-mail]
28. Date of birth: [Date]
29. Date of death: [Date]
30. Date of donation: [Date]

G_1	first ZZ number of loop 1	= 1
G_2	first ZZ number of loop 2	= C+1
G_3	first ZZ number of loop 3	= D+1
G_4	first ZZ number of loop 4	= E+1

The procedure is to check the first number of path 1 against all numbers of loop 1, then the second number of path 1, etc., until $YY_{A+1}(0)$ is found. If a number is found in the loop equal to the check number of the path, the loop touches the path (M1855), the loop number is increased by 1 and the check number of the path reset to YY_1 . If all numbers of the path are checked against a loop and no common point found, the loop does not touch the path and the loop number is recorded as the next subscripted XX, in this example the first loop found not to touch path 1 is recorded as XX_9 (M1900). After path 1 has been checked against all loops (M1835), the number of loops not touching path 1 is recorded as Y_1 (M1925). In the example, say that loop 1, 3 and 4 are found not to touch path 1. These are recorded as $XX_9=1$, $XX_{10}=3$, $XX_{11}=4$, $Y_1=3$.

To find the pairs of loops not touching path 1 and not touching each other, it is necessary only to check the loop numbers of single loops not touching path 1, against the loop pairs, recorded as XX_0 to XX_{2Z_1} . This procedure is indential to that used in the loop products routine:

(1)	(2)	(1)	(4)	(2)	(4)	(1)	(2)	(4)
XX_0	XX_1	XX_2	XX_3	XX_4	XX_5	XX_6	XX_7	XX_8

A

(1)	(3)	(4)	
XX_9	XX_{10}	XX_{11}	XX_{12} (not yet recorded)

Q	C	B	E
---	---	---	---

The variables A, B, E, Q are defined initially as the XX subscripts indicated and the recorded values of the XX variables are as indicated in parenthesis. The procedure is to check XX_Q against XX_A if not

equal, increase A by 2 and check again. In the example $XX_Q = XX_A$. The variable C then is defined as $Q+1$ ($=10$) and XX_C is checked against XX_{A+1} . These are not equal hence loops (1) and (2) are not a "pair not touching path 1". By the previously described numerical order of recording the XX variables, it is seen that if XX_C is greater than XX_{A+1} then further checks against XX_1 are necessary (M2035). The next pair is checked in the same manner. Again $XX_Q = XX_A$ (A now equals 2) and XX_C is checked against XX_{A+1} (M2025) and found unequal. Now, however, XX_C is less than XX_{A+1} so the possibility remains that XX_2 and XX_3 are a pair not touching this path. C is then increased by one (M2045) (to 11) and XX_C is again checked against XX_{A+1} . This time the two are found to be equal and XX_Q and XX_C are recorded as XX_{12} and XX_{13} . Now A is increased by 2 (to 4) and XX_Q is checked against XX_4 (M1965), found to be unequal and furthermore, XX_4 is greater than XX_Q , so no further test is necessary using XX_Q . The procedure continues and no more pairs are found. The number of pairs of loops not touching path 1 as recorded as Y_2 (M2100). Since only one pair has been recorded there can be no "triples not touching path 1". If more than one pair had been found, these would have been checked for triples in a similar manner to that used for loops and the number of triples is recorded as Y_3 (M2335). If none are found, Y_3 is recorded as zero.

The next step is to determine the end of the present path in the YY list (M2385, 2390), check to see if there are any more paths (M2400) and if so, repeat the process for the next path.

The entire XX list is summarized in the next paragraph. In addition to the XX variables, the following have been defined by this subroutine:

- Y_1 Number of loops not touching path 1
- Y_2 Number of pairs of loops not touching path 1
- Y_3 Number of loop triples not touching path 1
- Y_4
- Y_5 { Same as Y_1 , Y_2 , Y_3 applied to path 2
- Y_6

$$Y_3 Y_0^{-2}$$

$$Y_3 Y_0^{-1} \quad \{ \text{Same as above applied to the last path} \}$$

$$Y_3 Y_0$$

3.8. Performance Function Subroutine (M2500-3560)

Before discussing this subroutine, it is best to summarize the XX variables recorded in the routines of paragraphs 3.6., and 3.7:

$$XX_0 \text{ - - - - - Loop Pairs - - - - } XX_{2Z_1-1}$$

$$XX_{2Z_1} \text{ - - - - - Loop Triples - - - } XX_{2Z_1+3Z_2-1}$$

$$XX_{2Z_1+3Z_2} \text{ - - - - - Loop "4's" - - - - } XX_{2Z_1+3Z_2+4Z_3-1}$$

$$\text{Define } E = 2Z_1 + 3Z_2 + 4Z_3$$

$$XX_E \text{ - - - - Loops not touching } P_1 \text{ - - } XX_{E+Y_1-1}$$

$$XX_{E+Y_1} \text{ - - - Loop pairs not touching } P_1 \text{ - - } XX_{E+Y_1+2Y_2-1}$$

$$XX_{E+Y_1+2Y_2} \text{ - Triples not touching } P_1 \text{ - - - } XX_{E+Y_1+2Y_2+3Y_3-1}$$

$$\text{Define } F = E + Y_1 + 2Y_2 + 3Y_3$$

$$XX_F \text{ - - - - Loops not touching } P_2$$

.
.
.

etc. for all paths.

The last number recorded is XX_G , where G is defined as:

$$G = 2Z_1 + 3Z_2 + 4Z_3 + Y_1 + 2Y_2 + 3Y_3 + Y_4 + \text{ - - - } + 3Y_3 Y_0$$

As stated in Chapter 2., this routine is written to provide a performance function of the form

$$\frac{G_1 G_2 G_3 + G_4 G_5 + \text{ - - - }}{1 + G_a G_b G_c + G_d G_e + \text{ - - - }}$$

The results of the routine are the subscripts of the G's with appropriate algebraic signs. This routine is by-passed if not desired (M2501). The by-pass is effected as stated in paragraph 3.2., by entering a zero for the variable SS on the first data card.

The subroutine is quite naturally divided into two parts, namely, a Numerator and Denominator. Recall that the numerator is:

$$\sum P_k [1 + \sum L_{rk} + \sum \Pi L_{rk} + \sum \Pi L_{rk} + . . .]$$

In order to punch this function, it is necessary only to know where the particular values are stored in the computer. The method is very straight forward and will be described in detail only where particular sections of the program might require clarification.

The YY and ZZ lists, of path and loop components respectively, also contain the summers, which are not desired in the equation. Hence, any number in these lists greater than the number of components is not punched (M2600).

For convenience, the first YY number of each path is recorded (M2515 and M2635) in the same manner as the first ZZ number of each loop was recorded earlier. All components (excluding summers) of path 1 are punched successively (M2600, 2605) with no sign of positive and with a minus sign of the term of negative (M2585, 2595). The second term of the numerator $(P_k \sum L_{rk}) = P_1 L_{11} + P_1 L_{21} . . .$ is a series of Y_1 terms, the first of which contains all the components of path 1 and all the components of the first loop not touching path 1. This loop number is found in the XX list, the first ZZ number of the loop is known so the components of the loop are located and the entire term is punched.

Throughout the routine, there are many occasions to use a particular group of equations or commands, equations locating the number in the XX list, punch plus and punch minus commands, etc. To obviate the necessity of writing these same equations many times, two "routine indicators" P and R, are used to transfer control back to the desired program locations.

Continuing the numerator, the next series of terms is

$$P_k \sum_2 \prod L_{rk}$$

and the procedure is similar to that used before. After all terms have been punched, the same process is carried out for path 2 etc. until the entire numerator has been punched. To fully understand the mechanics of the program requires a step by step analysis of a sample system. This is done for a sample system of Chapter 4.

After all numerator terms have been punched, the program shifts to the denominator (M3083). Hence, the routine is simpler since the denominator of the performance function equation is

$$1 + \sum L_r + \sum_2 \prod L_r + \sum_3 \prod L_r + \sum_4 \prod L_r + \dots$$

The procedure is the same as for the loops of the numerator and the components of each loop, loop pair, etc. are punched successively with appropriate algebraic signs.

3.9. Frequency Response Calculation (M3570-M5285)

Having set up all terms of the performance function, there remains only the task of devising a scheme for entering the transfer functions of the system components, substitute $S = j\omega$, assign a frequency value and combine the results into the terms of the performance function. As described in Chapter 2, the component transfer functions are a ratio, the numerator and denominator of which are comprised of products of terms of the form $AS^2 + BS + C$, of which any of the coefficients may be zero. It is desired to enter on data cards only the coefficients. Since each transfer function may contain any number of terms, provisions must be made for recognizing the number of terms for each component. This is accomplished by means of an array of the subscripted variable G . For ease of entering the information a complete form is included with the sample problem of Chapter 4.

To understand the form in which the data is entered and the required information obtained, a simple example is shown in Fig. 3.2.

As stated previously, in order for the program to be properly executed, a scheme must be established whereby the terms of the array are recognized. The scheme chosen to accomplish this is to enter the G subscript of the first number of the last term of each numerator and denominator. These numbers will be referred to as "numerator stops" and "denominator stops" respectively. The numbers are shown circled in Fig. 3.2. and are entered in the following order:

- (1) Numerator stops listed consecutively, 6 to a card
- (2) Denominator stops listed consecutively, 6 per card.

The last card of each may not necessarily be full. In the example of Fig. 3.2., only one card is needed for each. The first card would contain: 0, 15, 24 and the second card: 6, 21, 27. These numbers are entered into the G-array immediately following the last number of the transfer function array, with a blank space between the two sets of numbers. This blank space is given the value of -3, which is necessary for manipulations later on.

Example of the Transfer Function Array

Consider a system of 3 components with transfer functions:

$$\begin{array}{l}
 1) \quad \frac{S + 1}{S (5S + 1)} \\
 2) \quad \frac{62 (.2S + 1) (3S + 1)}{(4S + 1) (S^2 + 7S + 1)} \\
 3) \quad \frac{3}{2S + 1}
 \end{array}$$

The coefficients are entered in the following order:

1. All numerator terms of component 1
2. All denominator terms of component 1
3. All numerator terms of component 2
4. All denominator terms of component 2
5. All numerator terms of component 3
6. All denominator terms of component 3

The numbers are entered consecutively, 6 per data card. These become a G-array as (numbers in parenthesis and circled are subscripts of the variable G):

	Constant		Coeff. of S		Coeff. of S ²	
Numerator 1	(0)	1	(1)	1	(2)	0

Denominator 1	(3)	0	(4)	1	(5)	0
	{ (6)	1	(7)	5	(8)	0
=====						
Numerator 2	(9)	62	(10)	0	(11)	0
	{ (12)	1	(13)	.2	(14)	0
	(15)	1	(16)	3	(17)	0

Denominator 2	(18)	1	(19)	4	(20)	0
	{ (21)	1	(22)	7	(23)	1
=====						
Numerator 3	(24)	3	(25)	0	(26)	0

Denominator 3	(27)	1	(28)	2	(29)	0

Fig. 3.2. G-Array of Coefficients of Transfer Functions

One other card is required to be entered and it contains the frequency information as: (M3570)

Initial frequency (radians /second) - - - - - VV

Frequency increment desired - - - - - DW

Final frequency (radians/second) - - - - - FW

Since a MAC READ order for reading in arrays of numbers must contain some information as to the number of cards to be read, it is necessary that the last G subscript of the transfer function array be entered. This number (shown circled in Fig. 3.2. as 29) is entered on the frequency data card and recognized by the computer as RR. (M3570).

Having read in the necessary data, the program picks up the transfer function of the first component. Letting $S = j\omega$ and substituting the initial frequency (VV), a magnitude and angle is computed for the component. Then the same process is carried through for each component. Referring to Appendix B, card M3640, the procedure is as follows:

1. Define the variable H initially as the subscript of the G variable left blank between the numerator and denominator stops.
2. Establish the variables K and L as the limits in the G-array of all terms (numerator and denominator) of component 1.

Space available as subscripted XX is used as temporary storage for the computation of the magnitude and angle of the components. The magnitude and angle of each term of component 1 is computed by M3690 and M3730. The command DO TO is used to accomplish this and means to carry out the operation specified for various values of the variable K from its present value in steps of (3) to a final value L. The operation is merely to determine the square root of the sum of the squares of the complex number representing each term of component number 1. The angle is found by M3730 as the ARCSIN of the imaginary part of the complex number divided by the magnitude. The ARCSIN function will give results between $\pm\pi/2$, hence, it is necessary to make the check of M3760 which states that if the real part of the complex number is negative, subtract the angle from π . If the real part is positive, the operation is ignored.

To see the manipulations more clearly, refer to Fig. 3.2.

$$R = 29$$

$$H = 33$$

$$K = G_{33} + 3 = 0$$

$$L = G_{34} = 6$$

This is seen to indicate 3 terms of component 1, and by M3690 and M3730, a magnitude and angle is computed for each of them. To obtain the magnitude and angle of the entire component, the numerator and

denominator terms are separated, each manipulated as necessary, then the magnitudes divided and the angles subtracted to give the results. This is accomplished by M3780 to M3925. Referring again to Fig. 3.2., and starting with M3785:

$$K = 0$$

$$M = G_{29+1} = 0$$

This means that there is only one term in the numerator and the numerator magnitude then is the magnitude already computed for that term. The numerator angle is found by adding the angles of all numerator terms (M3835, M3840) but in this case, there is only one.

A similar procedure is carried through for the denominator terms:

(M3855) $K = G_{28+3} = 3$ established S as the G subscript of the first denominator term. The two denominator terms are multiplied by M3865 and 3870, in two steps, $K = 3, 6$.

$$\text{M3850 } C = 1$$

$$\text{M3870 } C = (1) (XX_{E+3}) = XX_{E+3}$$

$$\text{M3870 } C = (XX_{E+3}) (XX_{E+6})$$

In a similar manner, the 2 denominator angles are added. (M3895, M3900)

It is desired now to compute a magnitude and angle for the component as a subscripted variable G_i in the interests of space economy, since later subroutines use the XX variable. Therefore, F is computed as the next available G number, and the component magnitude is computed by dividing numerator by denominator and identified as G_F . It is desired to have all component magnitudes in successive G numbers followed by all component phase angles. Hence, with Q components the phase angle of component one is computed by subtracting denominator angle from numerator angle and identifying the result as G_{F+Q} .

The procedure now is to return to the beginning of this subroutine and carry out the above process on the remaining components in succession. Since the component magnitude and phase angles will be used

later, it is convenient to define the variable J (M3970) as the last G number preceding the first component magnitude. Then to find the magnitude of component N, the location is known to be G_{J+N} .

It would be satisfactory now to substitute these components magnitudes and phase angles into the performance function as punched in paragraph 3.8., perform the required algebra and obtain the response of the system for this frequency. However, the performance function is not retained as such and calculation of the performance function is desired to be optional. Since the XX variables are stored as loop numbers, each loop number appearing many times, a more convenient and faster method is to compute the magnitude and phase angle of each loop. Then the loop response is used in computing the response of the loop product terms. All of these are terms in the performance function and will be referred to herein as "term responses".

The first step is to devise a storage scheme for these term responses so they may be easily combined. The denominator of the performance function is computed first and the storage scheme used is (M4005 - 4025): First available XX variable: XX_E

$XX_{E+1} - - - - - XX_K$	$XX_{K+1} - - - - - XX_L$
Loop magnitudes	Loop phase angles
$XX_{L+1} - - - - - XX_C$	$XX_{C+1} - - - - -$
Real part of all term responses	Imaginary part of all term responses.

The denominator is a summation of term responses:

$$(1 + \sum L_r + \sum \prod_2 L_r + . . .)$$

After computing these term responses and converting to rectangular form, all that is necessary is to add the real parts (XX_{L+1} to XX_C) and the imaginary parts (XX_{C+1} to $XX_{C+Z_0 + Z_1 + Z_2 + Z_3}$), to convert

the result to polar form and retain it until after the numerator response is computed.

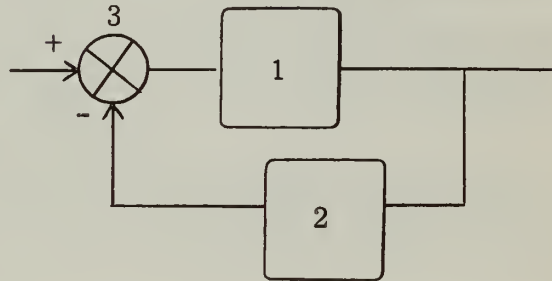
In computing loop magnitudes recall that the ZZ list of loop components (paragraph 3.4.) has a sign determined by the number of minus signs encountered. For example, a single loop is recorded as:

$$ZZ_1 = -3$$

$$ZZ_2 = 1$$

$$ZZ_3 = 2$$

$$ZZ_4 = 0$$



As explained in Chapter 2 and from elementary feedback theory the sign of this loop is (+). Hence, an initial multiplier of -1 is used (M4045) and an initial phase angle of 0 (M4050). Component magnitudes and phase angles have been stored as G_{J+A} and G_{J+A+Q} respectively. The loop magnitude and phase angle is computed (M4055 - 4115) using only components (summers are excluded), with the appropriate sign, and recorded as described in the storage scheme above. It is required that the loop response be retained in both polar and rectangular form (M4130 - 4160).

The response of the terms representing loop pairs is obtained by (M4210 - 4290) using the loop numbers of the pairs from XX storage beginning with XX_0 and multiplying magnitudes and adding phase angles of the loops. Since these loop pair responses are never multiplied in the performance function equation, they are converted to rectangular form (M4250 - 4260) and retained only in this form.

By exactly the same method, the term responses for the loop "triples" and "4's" are computed and retained in rectangular form (M4310 - 4500). The denominator response is then obtained by adding all real parts of the term responses, stored consecutively as XX_{L+1} to XX_C , and all the imaginary parts, XX_{C+1} to XX_{C+T-1} . T is a variable used to increase the XX variable by one each time a new term was added, hence it can be used as a check to determine the end of the

list to be added. There are T-1 numbers in each list. The result of the additions, a complex number, is converted to polar form (M4580-4630) and retained as denominator magnitude, (DM) and denominator phase angle (DA).

Note, that the conversion from rectangular to polar form is a routine that is to be used more than once, so the routing identifier (X) is used (M4570). As noted previously, the ARCSIN function given only angles between $\pm \frac{\pi}{2}$, so if the summation of the real part is negative the correct angle must be computed (M4605).

Obtaining the numerator response is only slightly more complicated since it is a summation of terms multiplied by a path function and then the products are summed:

$$\sum P_k [1 + \sum L_{rk} + \sum \frac{\pi}{2} L_{rk} + \dots]$$

The procedure is to determine the path response (P_k) for path 1, in polar form, then determine the term response of each of the terms indicated in the equation that are associated with this path, perform the addition indicated in the brackets and convert to polar form. This result is then multiplied by the path response, the final result converted to rectangular form and stored.

The procedure is then repeated for the second path, and so on. After all paths have been completed the final summation, $\sum P_k [1 + \dots]$ is performed and the result converted to polar form.

The loop responses previously computed are needed here but the other denominator terms are not, hence, the same storage may be used for the numerator terms. In the program of Appendix B, this is only partially true, in that it was chosen to use only the space after XX_{C+Z_0} :
($B = C + Z_0$)

XX_{E+1} - - - - -	XX_{L+1} - - - - -	XX_{C+1}
Loop magnitude and phase angle	Real part of all denominator term responses	Imaginary part of loop responses
XX_{B+1} - - - - -	XX_{H+1} - - - - -	
Real part of all numerator term responses.	Imaginary part of all numerator term responses.	

The magnitude and phase angle of the first path are computed (M4690- M4760) in the same manner as that of the loops except that the sign of a path function is minus if the path in the YY list has a minus sign. The path magnitude and phase angle are retained temporarily as V and W respectively. (M4720 - 4730).

The numerator terms associated with the path are located in XX storage from the storage scheme summarized in paragraph 3.8. The first group of these terms ($\sum L_{rk}$) is a summation of loops for which the real and imaginary parts of their response have been previously computed, so need only be rewritten as the appropriate XX variable (M4770 - 4825).

The second group of numerator terms ($\sum \prod_2 L_{rk}$) and the third group ($\sum \prod_3 L_{rk}$) are determined exactly as the $\sum \prod_2 L_r$ and $\sum \prod_3 L_r$ terms of the denominator (M4835 - 4915 and M4925 - 5015).

Having determined the response of all terms associated with this path, their real parts + 1 and imaginary parts are summed (M5030-5085). The routing identifier X = 1 is used and the program jumps to the routine for converting rectangular to polar form (M5095).

Up to this point, the following have been obtained: $V/W [P/A]$ (for $1 + \sum L_{rk} + \dots$). The response for all terms associated with this path is then computed (M5100 - 5120) and retained as an XX variable. If any more paths exist (M5130) the same routine is carried out for each.

After all paths have been completed, the real and imaginary parts are summed (M5155 - 5195). The storage locations of these values are:

$$\begin{array}{ccccccc} XX_C & - & - & - & | & XX_{B+1} & XX_{B+2} & | & XX_{B+3} & XX_{B+4} & | & \text{etc.} \\ & & & & & \text{first path} & & & \text{second path} & & & \end{array}$$

Each time a path was completed B was increased by 2 in preparation for working space of the next path. After all paths are completed, B is set back to its original value (M5156) to permit summing.

The program now jumps to the conversion of rectangular to polar form routine (M5210), the result of which gives the numerator magnitude and numerator phase angle. The system magnitude ratio (M) and phase angle (P) are found (M5215 - 5220). The final result is desired with the phase angle given as a positive angle between 0 and 360 degrees, hence, the two equations (M5235 and 5255). The possibility exists that M might be zero or the DM might be zero (M would be infinite). In either case, special punch orders are provided (M5212, 5275 and 5237, 5253).

The following information is obtained from the punched result:

M	Magnitude Ratio
P	Phase Angle in Degrees (0-360)
AA	Log M (MAC computes logs to the base e so it is converted to the base 10)
20AA	Magnitude Ratio in d.b.
VV	Frequency (radians/sec)
VV/2 π	Frequency (cps)

3.10. Special Features

There are certain special situations that are not strictly a part of any of the general subroutines of the preceding paragraphs, but might require explanation. One such situation might arise when the system performance function is known in factored form, perhaps from a pole-zero configuration. In this case, the block diagram of the system is:



Component 1 may consist of any number of terms. There are no loops and only one path. The calculation of frequency response of the system is made treating the system as one component and the entire program up to and including the performance function routine is by-passed. This by-pass is effected by the presence of a zero on the first data card in the position designated as "number of summers" (M0035, 0036). The number of loops (Z_0) is recorded as zero in the by-pass (M2565) and the variable E, used to establish storage locations in the XX list is

assigned a value of zero.

The "input data array" is not entered. The first card must contain six numbers since the READ Order (M0035) says read six values. However, the only ones with any meaning in this case are the third, where a zero is entered for S, and the second where the number of components is entered as 1. The remainder of the data cards are entered (M3570, 3585, 3595, 3605) exactly as explained in paragraph 3.9.

A second special situation arises when, having computed a frequency response, it is desired to change the frequency range, frequency increment, or a transfer function of one of the existing components. In this case, it is desired that all of the loops, paths, and combinations leading to a performance function be retained, and only that part of the program pertaining to the calculation of frequency response be repeated. A simple procedure for handling this is to let the program run to completion, have new data cards available, and start again at M3570. The program is set up in such a manner that all that is required is to push the start switch and the program will start at M3570.

If in addition to the above, it is also desired to insert a new component or change the configuration of the existing circuit, it is still not necessary to read the entire load deck again. A "program start" may be entered directly into the computer in basic machine language. This entails setting up on an "index register", the address of the appropriate command in the computer. This is a minor task but to ensure proper operation, a detailed step-by-step procedure follows:

Detailed Step-by-Step "Re-run" Procedure

1. Prepare data cards as in original situation.
2. Set 1698 in the ADDRESS SELECTION SWITCHES on the main console. These switches are plainly labeled.
3. Set computer CONTROL SWITCH TO MANUAL.

4. Press PROGRAM RESET key.
5. Press TRANSFER key.
6. Set CONTROL SWITCH to RUN.
7. Press PROGRAM START key.

CHAPTER 4

SUMMARY OF PROCEDURES FOR USING THE PROGRAM

4.1. General

In Chapter 2, the general plan of the program was discussed with the intent of describing its application to any computer. Chapter 3 covered the details of the program, of Appendix B, written in MAC language for the IBM 650 and its associated equipment. The mechanics of the program and the method used to obtain the frequency response have been adequately discussed. In this chapter, a summary of the procedures required by the user will be outlined with application to a specific example.

It is noted that MAC uses plain language for the basic program but that this program is then assembled in "machine language" prior to operation. The "load deck" referred to in Chapter 2 is a machine language version of the MAC program. Any other information entered in the form of data cards must be in a form recognized by the computer and independent of MAC. Data cards contain only 10-digit numbers, six or less to a card, and the numbers are in "floating point" form. Floating point is merely a method of placing the decimal point in a number always written with the first digit as non-zero. The first 8 digits are the numerical value and the last two locate the decimal point. For a number with the decimal point at the extreme left (e.g. .62), the 9th and 10th digits are 50. For each digit that the decimal point is located to the right or left of the first non-zero digit of a number, 1 is added or subtracted from 50. For example:

62.0	is written in floating point as	62000000 52
6.20	as	62000000 51
.620	as	62000000 50
.062	as	62000000 49

4.2. Data Cards Required

The data cards are placed in the card reader feed directly on top of the MAC or load deck and are read at the appropriate time by a READ order in the program. The number of data cards required will vary with the complexity of the physical system being analyzed. However, the type of data falls into six general groups. A new card must be punched at the beginning of each of these groups since they are read in by different READ orders. The last card of each group is not necessarily filled. The six general groups and the data cards of each are as follows:

(a). General Circuit Information (1 card)

- (1) Maximum number of inputs to any unit. K
- (2) Number of component boxes in the block
diagram of the system. QQ
- (3) Number of summers. S
- (4) Number of the input unit CC
- (5) Number of the output unit DD
- (6) Is Performance Function desired? . . . SS = 0 if No
1 if Yes

(b) Input Data Array (M cards)

(where $M = QQ + S + 1$, unless K is greater than 4, in which case, the number of values on each card exceeds 6 and additional cards must be punched)

The "input data array" described in paragraph 2.2. and 3.3. is entered, one row per card (unless the row contains more than 6 numbers, in which case, an additional card is used for each multiple of 6). Each row of the array is started on a new card. The last row of the array and hence the last card contains all zeros.

(c) Frequency Data (1 card)

- (1) The lower limit of the frequency range over
which the response is desired VV
- (2) The increment of frequency over the fre-
quency range DW

- (3) The upper limit of the frequency range . . . FW
- (4) The last subscript number of the G-array
of Transfer Functions RR

Note that only four numbers on this card will be read. The last two spaces will not and zeros may be entered in these spaces if desired:

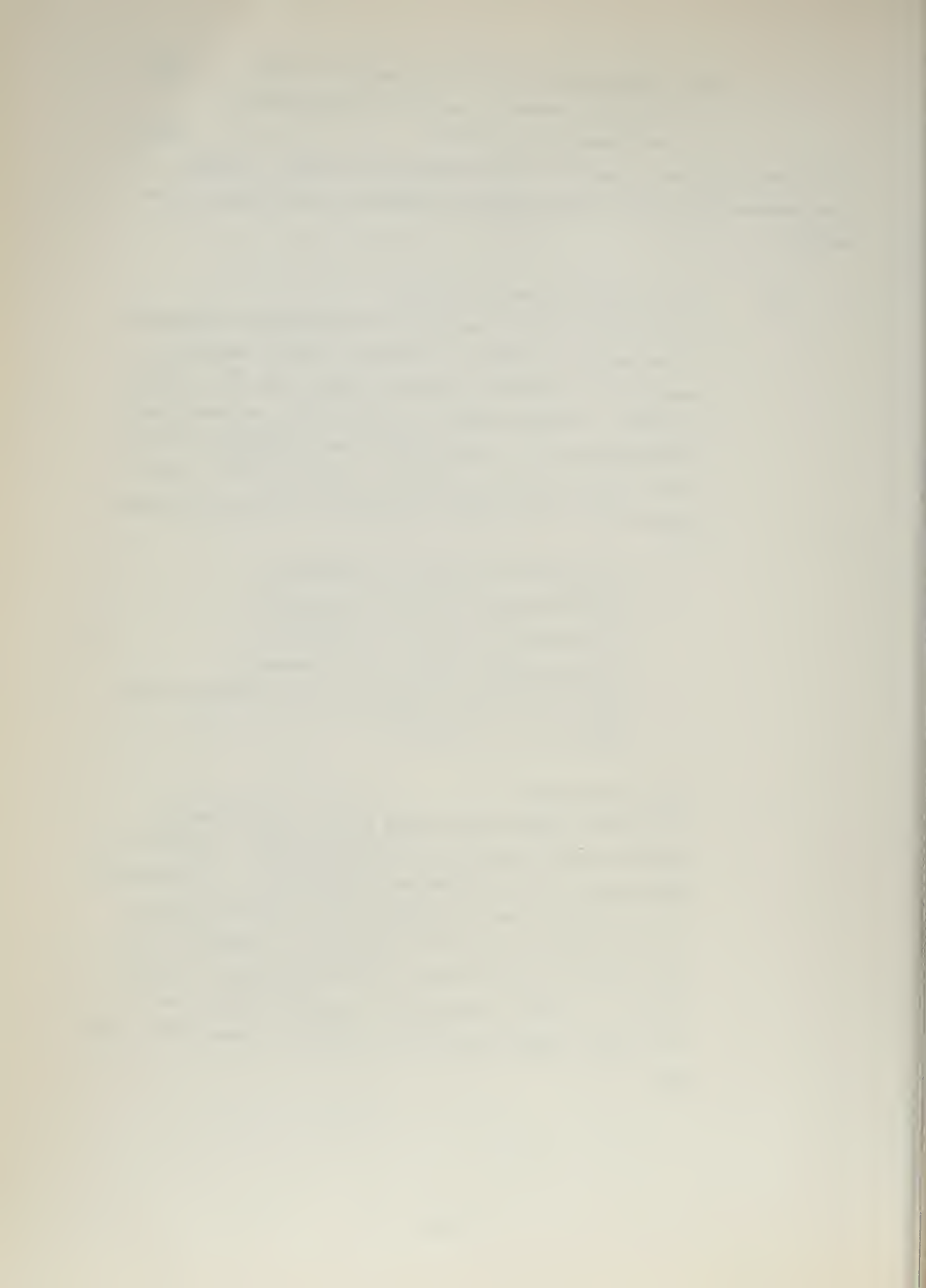
d. The Transfer Function Array

The number of cards here will vary with the complexity of the physical system. Numbers on these cards are punched in succession, six to a card, until all transfer function coefficients have been punched. As described in paragraph 3.10., the order of the coefficients entered for each term is: constant, coefficient of S , coefficient of S^2 . The order in which the terms are entered is:

- All numerator terms of component 1
- All denominator terms of component 1
- All numerator terms of component 2
- All denominator terms of component 2
- And so on for each component; the last card may contain 3 or 6 numbers.

e. Numerator Stops

A new card is started for this group, regardless of whether the last card of group (d) is filled. Numbers on these cards, as explained in paragraph 3.9., represents the subscript of the G number in the transfer function array, containing the first coefficient (constant) of the last term of the numerator of each component transfer function. These numbers are punched consecutively, 6 to a card, except that the last card may contain less than six.



(f) Denominator Stops

A new card is started for the first numbers of this group. As explained in paragraph 3.9., numbers on these cards represent the subscript of the G number in the transfer function array, containing the first coefficient (constant) of the last term of the denominator of each component transfer function. These numbers are punched consecutively, 6 to a card.

4.3. Results of the Program

As described in Chapter 3 and illustrated by the example in this chapter, the result of the performance function routine (if such is desired) is a series of groups of numbers representing the component making up a term in the performance function equation. These groups of numbers (terms of the equation) are preceded by the appropriate algebraic sign.

Results of the frequency response calculation are given, for each frequency, as a row of six numbers, under proper heading and represents:

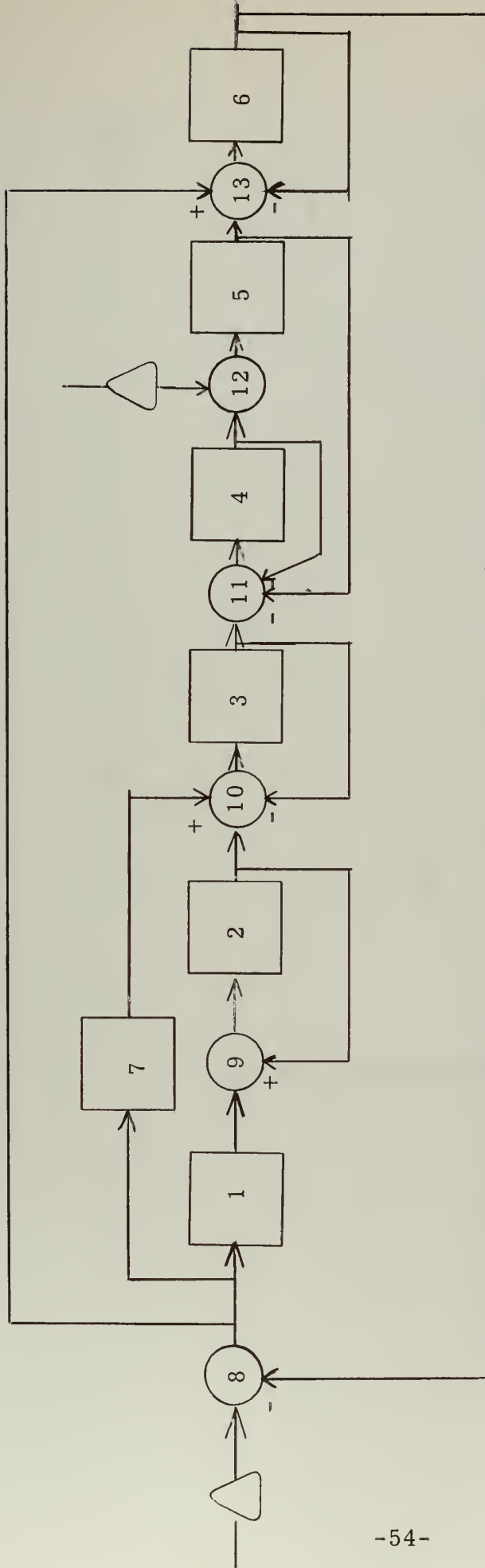
- a. Magnitude ratio
- b. Phase angle (in degrees)
- c. Log of magnitude ratio
- d. Magnitude ratio in decibels
- e. Frequency (radians/sec)
- f. Frequency (cycles/sec)

4.4. Example

Consider the system represented by the block diagram of Fig. 4.1. with transfer functions as indicated. This is a hypothetical system chosen of sufficient complexity to adequately test all parts of the program and to illustrate the advantage of digital computation over graphical techniques.

The first step is to list the information required by the single card





1	→ 15	3	→ $\frac{10}{S(S+1)}$	5	→ 5
2	→ $\frac{1}{S(S+1)}$	4	→ $\frac{.2S+1}{S^2+S+1}$	6	→ $\frac{1}{.5S+1}$
				7	→ $\frac{1}{.03S+1}$

Fig. 4.1. Block Diagram of System
Used for an Example



Output ↓	Inputs			
1	8	0	0	0
2	9	0	0	0
3	10	0	0	0
4	11	0	0	0
5	12	0	0	0
6	13	0	0	0
7	8	0	0	0
8	-6	0	0	0
9	1	2	0	0
10	2	-3	7	0
11	3	-4	-5	0
12	4	0	0	0
13	5	-6	8	0
0	0	0	0	0

Fig. 4.2. Input Data Array



	G. Subs.	Const.	G. Subs.	Coef. of S	G. Subs.	Coef. of S ²
1	(ϕ)	15	1	0	2	0
	(3)	1	4	0	5	0
2	(6)	1	7	0	8	0
	9	0	10	1	11	0
	(12)	1	13	1	14	0
3	(15)	10	16	0	17	0
	18	0	19	1	20	0
	(21)	1	22	1	23	0
4	(24)	1	25	.2	26	0
	(27)	1	28	1	29	1
5	(30)	5	31	0	32	0
	(33)	1	34	0	35	0
6	(36)	1	37	0	38	0
	(39)	1	40	.5	41	0
7	(42)	1	43	0	44	0
	(45)	1	46	.03	(47) ↓ RR	0

Fig. 4.3. Transfer Function Array





of group (a). Values for the variables are seen to be:

K = 3
QQ = 7
S = 6
CC = 8
DD = 6
SS = 1

Succeeding steps in preparing data for data cards are:

1. Write the input data array as in Fig. 4.2.
2. List the frequency data. For this example, a frequency range of .1 to 6.0 radians per second was chosen with a frequency increment of .4 radians/sec. Note, that the last item on this card does not pertain to frequency data but is a required entry and placed on this card for convenience. Its value will be determined later.
3. Write the transfer function array as in Fig. 4.3. Indicate in some manner (circles are used in Fig. 4.3) the numerator and denominator stops and the last G subscript of the array.

Data cards are now punched in floating point form. These are illustrated in Fig. 4.4.

The program, followed by the data cards, is then placed in the card reader and the computer set in operation. Since the first data card indicated a performance function was desired, the first result is this performance function. For this fairly complicated system, the performance function is quite lengthy so is appended as Appendix C.

The frequency response results are listed and plotted as shown in Appendix D. Note, that a plotter is available and when set to the appropriate scales will produce a plot directly from the punched cards. The plotter was not used for this example.

4.5. Summary and Recommendations

This report has illustrated the basic simplicity of the plan and the feasibility of programming such a plan for digital computation.

the first of these is the fact that the
 the second is the fact that the
 the third is the fact that the
 the fourth is the fact that the
 the fifth is the fact that the
 the sixth is the fact that the
 the seventh is the fact that the
 the eighth is the fact that the
 the ninth is the fact that the
 the tenth is the fact that the

the first of these is the fact that the
 the second is the fact that the
 the third is the fact that the
 the fourth is the fact that the
 the fifth is the fact that the
 the sixth is the fact that the
 the seventh is the fact that the
 the eighth is the fact that the
 the ninth is the fact that the
 the tenth is the fact that the

the first of these is the fact that the
 the second is the fact that the
 the third is the fact that the
 the fourth is the fact that the
 the fifth is the fact that the
 the sixth is the fact that the
 the seventh is the fact that the
 the eighth is the fact that the
 the ninth is the fact that the
 the tenth is the fact that the

Some salient features are:

- a. The limit on size and complexity of the system to be analyzed is that caused by capacity of the computer, more specifically, the capacity of the particular method chosen.
- b. The time required for analysis is:
 1. Read load deck - - 5 minutes
 2. Computation of all the terms of the performance equation. This will vary with the complexity of the system. For the system used in example number 1, the time was approximately 8 minutes.
 3. Computation and punching the performance function. Again this time will vary with the system. For example number 1 the time was approximately 4 minutes.
 4. Computation of frequency response for the system of example 1 - - slightly under 2 minutes per frequency point.

As a future project it is recommended that:

- a. The program be refined to reduce computation time.
- b. The refined program be altered to make use of auxiliary equipment such as tape to lessen present physical limitations.
- c. With greater storage capacity, the performance function routine be extended to enable computation of the performance function as a ratio of polynomials of S .
- d. Having accomplished (c), an investigation be made into the feasibility of extending the program to include factoring the polynomials and obtaining a time response.



APPENDIX A

Program Summary No. 1, L_r Subroutine (M0075 - M0585)

The purpose of this summary is to explain the details of the program covering the method explained in paragraph 3.4. Using the G-array of paragraph 3.3., the first summer listed is located by knowing the number of components and the "width" of the array. For the sample system of paragraph 3.3., this starting point is G_{20} , the value of which is (6). As in paragraph 3.4., the G-array is searched by rows, starting with G_1 (encountering a zero indicates the end of the row) until the search number is found. It is located in G_5 . To find the associated output (first number of the row containing G_5) the equation: $L = M \text{ INTEGER } (L/M)$ is used; M is the width of the array and L is the G subscript (5). A new value of L is computed then as (4) hence, the number wanted is $G_4 = (2)$. Continuing as in paragraph 3.4., no more outputs of (6) are found. The (2) is used as a search number and is found to go to (5) and also negatively back to (7).

Before proceeding further, it is best to look at the XX-array. (The numbers in parenthesis indicate the subscript of XX locating the position in the array).

XX-ARRAY

(0)		
6		
(1)	(2)	
6	2	
(3)	(4)	(5)
6	2	5
(6)	(7)	(8)
-6	2	7

Note, that the (-) appears on the first number of the line rather than on the number that caused it. The command that finds the search number in the G-array (M0230 and M0240) checks for both positive and negative numbers and if negative sets a variable R = 1 (0255). Then when the line is recorded, each term is multiplied by R (M0350). After the first number of the line is computed, R is set to +1. This is desirable since in later computations, it is the sign of a loop that is important, so the sign of the first number only is checked.

The method of this routine is adequately described in paragraph 3.4. The following is concerned primarily with the mechanics of the program. As noted earlier, before recording a line in the XX-array certain checks must be made, the first of which is: "Check of a complete loop (M0265)". This checks the next number to be recorded against the first number of the line and if they are the same, the numbers of that line represent units comprising a complete loop and are placed in the ZZ list (M0500). The next check is to see if the point is already recorded in the appropriate line. The newly found number must be checked against all numbers of the line (M0275 to M0285). If it is a duplicate point, a new line is not recorded but the search is continued. If not a duplicate point, then it must be checked against all summers previously used as starting points; these are recorded by (M0485) and the check is made by (M0320).

The mechanics of recording lines in the XX-array is fairly complicated and is most easily understood after getting farther along in the array, although the same commands are used to record the first few lines. Consider the stage of operation in paragraph 3.4., at which is recorded:

	(9)	(10)	(11)	(12)	
line (e)	-6	2	7	3	(XX ₉ to XX ₁₂)
	(13)	(14)	(15)	(16)	
line (f)	-6	2	7	4	(XX ₁₃ to XX ₁₆)
	(17)	(18)	(19)	(20)	
line (g)	-6	2	5	8	(XX ₁₇ to XX ₂₀)



		(21)	(22)	(23)	(24)	(25)	(XX ₂₁ to XX ₂₅)
	line (h)	-6	2	5	8	1	
(4)		(26)	(27)	(28)	(29)	(30)	
	line (i)	-6	2	7	4	8	(XX ₂₆ to XX ₃₀)
		<hr/>					
		(31)	(32)	(33)	(34)	(35)	(36)
(5)	line (j)	-6	2	7	4	8	1 (XX ₃₁ to XX ₃₆)
	line (k)						

The array is separated into "blocks" by the solid lines and the blocks are given numbers indicated by the number to the left of the array.

To handle this recording procedure, the following variables are used:

W	Number of current block (5)
Y	The XX subscript of the first number of the last line of the previous block
V	The XX subscript of the first number of a line containing the present search number.
FF	Present search number
N	In general, the XX subscript of the first number of the last recorded line.

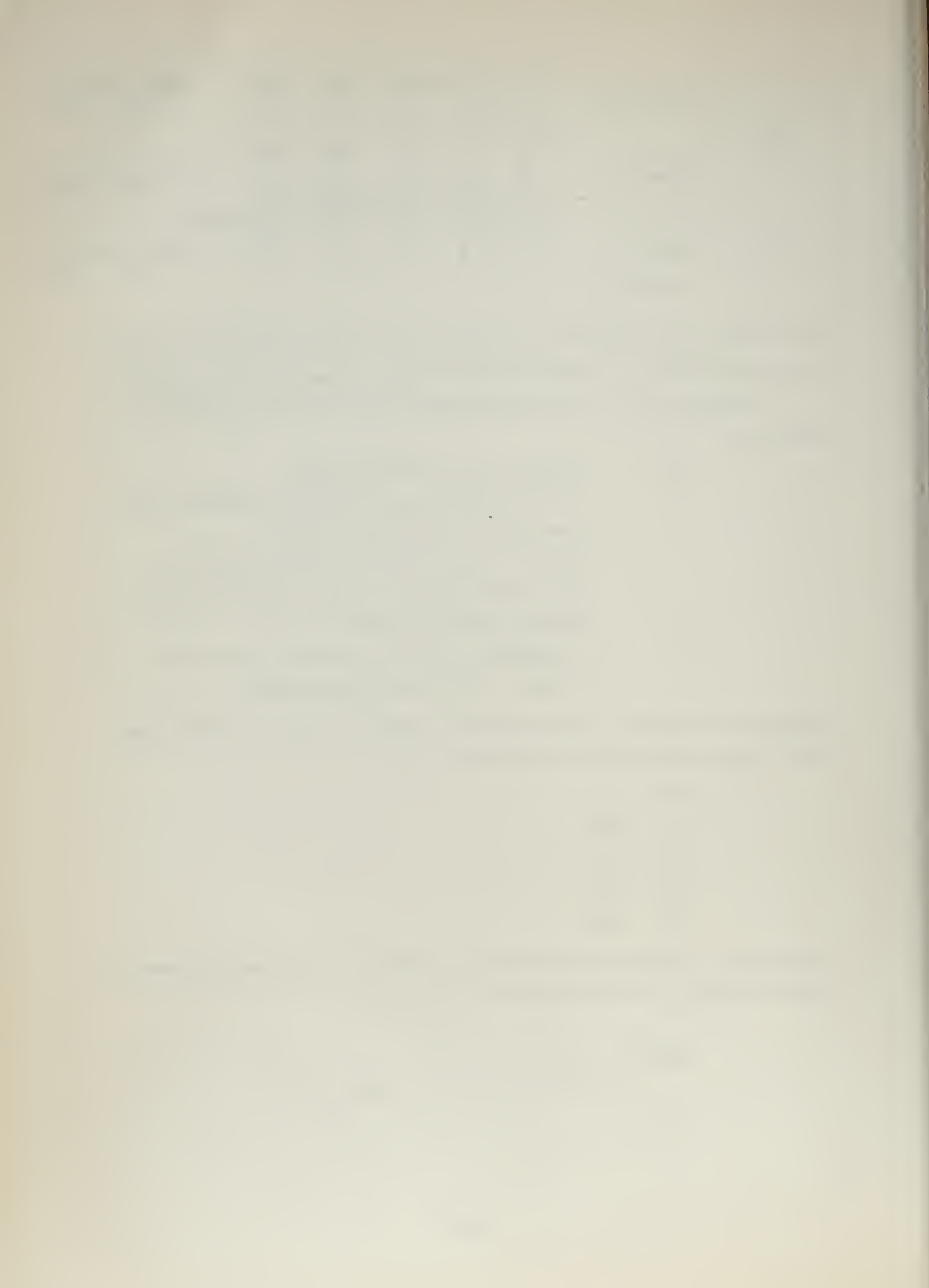
From the array above, assume line (j) has just been recorded using $FF = XX_{30} = (8)$ as a search number; then

W	= 5
V	= 26
N	= 31
Y ₄	= 17
Y ₅	= 26

Returning to search, no more outputs of (8) are found and it is desired to pick up $XX_{25} = (1)$ as the new value of FF.

$$M0400 \quad V = V - W = 21$$

$$FF = XX_{V + W - 1} = XX_{25} = (1)$$



Return to search the G-array and find (-1) going to (7) and after making the 3 checks noted previously increase N to $N + W + 1 = 37$ and record (M0350) line (k) as

+6, 2, 5, 8, 1, 7 (XX₃₇ to XX₄₂)

No more outputs from (1) are found and a new search number must be picked up. From the array above, it is seen that it is not desired to go back another line but to start a new block. The reason for the Y() variable is now apparent. Change V, as before, to $V = V - W = 16$ and check it against Y_{W-1} to see if block 4 has been finished (M0405).

$$V = N = 37$$

$$FF = XX_V + W = XX_{42} = 7$$

$$W = W + 1 = 6$$

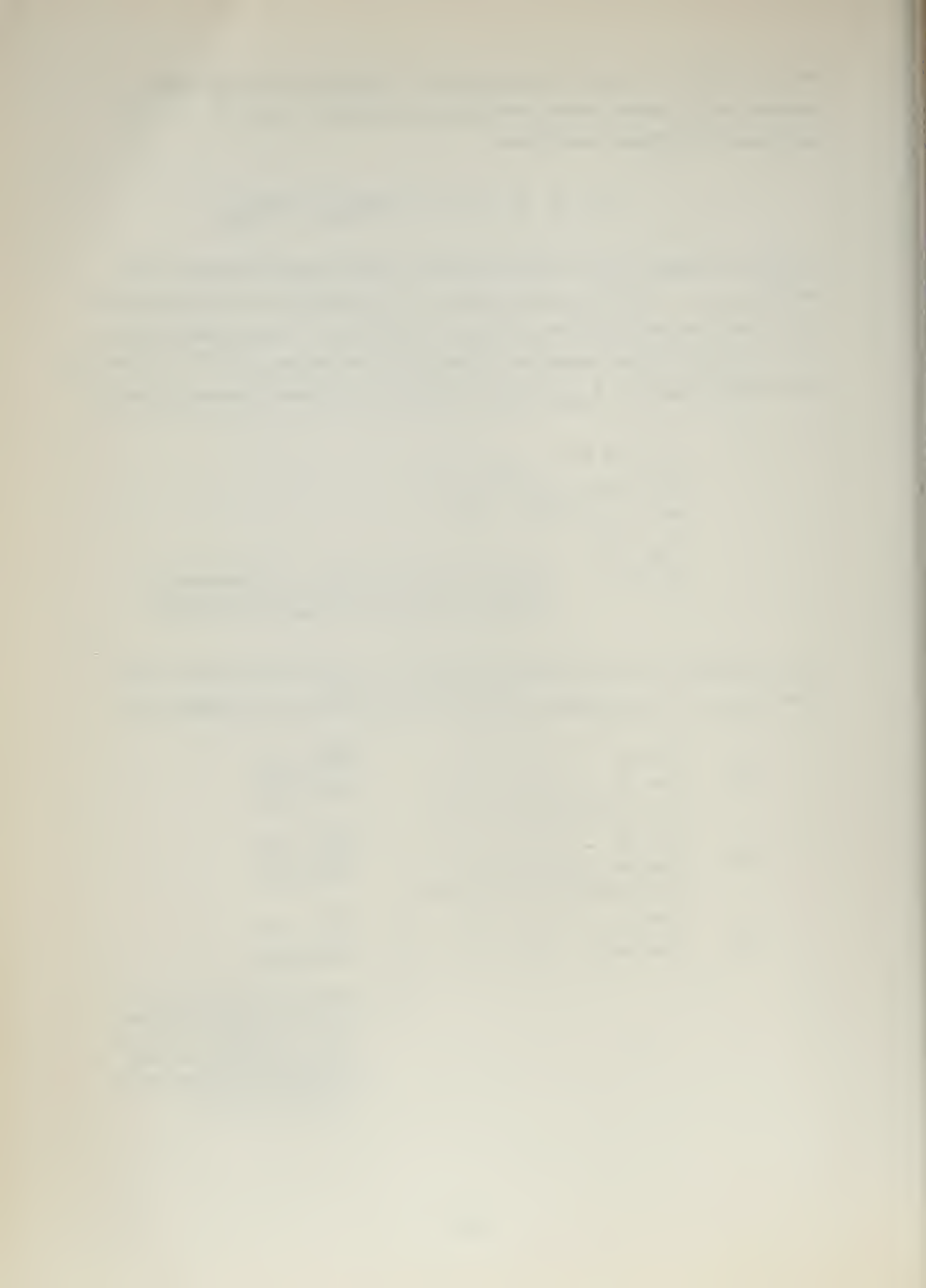
$$Y_6 = 37$$

N = N - 1 (The reason for this is not apparent at present but will be explained shortly)

The variables are now set up to go back to search and complete another block. The process continues until the XX-array appears as

(4)	line (h)	-6 2 5 8 1	(XX ₂₁₋₂₅)
	line (i)	-6 2 7 4 8	(XX ₂₆₋₃₀)
(5)	line (j)	-6 2 7 4 8 1	(XX ₃₁₋₃₆)
	line (k)	+6 2 5 8 1 7	(XX ₃₇₋₄₂)
(6)	line (l)	6 2 5 8 1 7 3	(XX ₄₃₋₄₉)
	line (m)	6 2 5 8 1 7 4	(XX ₅₀₋₅₆)

* Note that nothing from line (j) is recorded here since (1) goes to (6) and forms a complete loop; also goes to (7) which is a duplicate point.



Having recorded line (m) and set up to start a new block, the values of the variables are:

$$V = N = 50$$

$$FF = 4$$

$$W = W + 1 = 7$$

$$Y_7 = 50; Y_6 = 37$$

$$N = N - 1 = 49$$

Return to search with $FF = 4$ and find it goes only to (8), a duplicate point.

$$V = V - W = 43$$

$$FF = XX_{V+W-1} = 3$$

Return to search and find (3) goes only to (6), a complete loop, so nothing is recorded.

$$V = V - W = 36$$

V is less than Y_6 hence it is desired to start a new block but the routine, for this summer, is finished since there is nothing recorded in block (7). Recall that if anything had been recorded in block (7), N would have been changed to $N + W + 1 = 57$. Now, when V is set equal to $N = 49$, the check against Y_W indicates the end.



APPENDIX B-1
CLOSED LOOP ROUTINE

```

M0005 INDICES A,B,C,D,E,F,H,K,L,N,P,
M0010 Q,S,T,U,V,W,R,J,M
M0015 RESERVE G,XX,YY,ZZ
S0020 220 320 75 100

M0025 ,BB,Y
S0030 10 25

M0035 READ K,QQ,S,CC,DD,SS 16 READ INPUT CONSTANTS
M0036 IF S ZERO, GO TO 190 ARE THERE ANY SUMMERS
M0037 Q=QQ

M0040 M=Q+S+1 NUMBER OF DATA CARDS
M0045 P=0

M0050 READ G TO G 21 READ CIRCUIT ARRAY
S0055 P P+K+1
M0060 P=P+K+2
M0065 M=M-1
M0070 IF M NZ, GO TO 21
M0075 BB =0
S0080 0 BEGIN LR ROUTINE

```



APPENDIX B-1
CLOSED LOOP ROUTINE

M0085	M=K+2	
M0087	Z = 0	
S0088	0	
M0090	U=M(Q-1)	SET FIRST NODE NUMBER
M0095	H=0	
M0096	T=1	
M0100	U=U+M	
M0105	N=-1	2 INCREASE NODE NUMBER
M0110	P=0	
M0115	R=1	
M0120	S=1	
M0130	V=0	
M0145	W= 1	
M0150	XX =G	
S0155	0 U	



APPENDIX B-1 CLOSED LOOP ROUTINE

M0160	Y =0		
S0165	I		
M0170	Y =0		
S0175	0		
M0180	FF=G		SET FIRST NODE SEARCH POINT
S0185	U		SET FIRST NODE START POINT
M0190	IF G ZERO, GO TO 18		TEST FOR COMPLETION
S0195	U		OF LR ROUTINE
M0200	L=0	3	SET FIRST SEARCH NO.
M0205	L=L+1	19	INCREASE SEARCH SUBSCRIPT
M0210	IF G ZERO, GO TO 15		TEST FOR END OF ROW
S0215	L		
M0220	IF G ZERO, GO TO 23	4	TEST FOR END OF COLUMN
S0225	L		
M0230	IF FF-G ZERO, GO TO 5		TEST FOR POSITIVE VARIABLE
S0235	L		
M0240	IF FF+G ZERO, GO TO 14		TEST FOR NEGATIVE VARIABLE
S0245	L		



APPENDIX B-1
CLOSED LOOP ROUTINE

M0250	GO TO 19		
M0255	R=-1	14	NOTE NEGATIVE VARIABLE
M0260	L=M INTEGER(L/M)	5	ASSOCIATED OUTPUT
M0265	IF ABS(XX)-G		
S0270	V+P L		TEST FOR END OF LOOP
M0275	P=P+1	6	
M0280	IF W-P ZERO, GO TO 10		
M0285	IF XX -G NZ, GO TO 6		TEST FOR DUPLICATE POINT
S0290	V+P L		
M0295	L=L+M+1	9	
M0300	P=0		
M0305	R=1		
M0310	GO TO 4		RETURN TO SEARCH
M0315	P=H	10	
M0320	IF BB -G ZERO, GO TO 9		
S0325	P L	11	TEST FOR PREVIOUSLY COMPLETED NODE



APPENDIX B-1
CLOSED LOOP ROUTINE

```

M0330 P=P-1
M0335 IF P PNZ, GO TO 11
M0340 P=0
M0345 N=N+W+1
M0350 XX =RXX
S0355 N+P V+P
M0360 R=1
M0365 P=P+1
M0370 IF W-P NZ, GO TO 8
M0385 XX =G
S0390 N+P L
M0395 GO TO 9
M0400 V=V-W
M0405 IF V-Y NEG, GO TO 25
S0410 W-1
M0415 FF=XX

```

8 RECORD NEW XX ARRAY LINE

RECORD LAST NUMBER OF XX ARRAY LINE

RETURN TO SEARCH

23

TEST FOR END OF XX ARRAY BLOCK

SET NEW SEARCH NUMBER



APPENDIX B-1
CLOSED LOOP ROUTINE

S0420	V+W-1	
M0425	GO TO 3	
M0430	V=N	25 SET NEW LINE FOR XX ARRAY BLOCK
M0435	FF=XX	SET NEW SEARCH NUMBER
S0440	V+W	
M0445	IF V-Y NEG, GO TO 24	TEST FOR END OF THIS NODE XX ARRAYS
S0450	W	
M0455	W=W+1	
M0460	Y =N	
S0465	W	
M0470	N=N-1	
M0475	GO TO 3	
M0480	H=H+1	24
M0485	BB =G	RECORD COMPLETED NODE
S0490	H U	
M0495	GO TO 2	


```

12 RECORD LOOP COMPONENTS
-
M0500 ZZ =RXX
S0505 T+P V+P

M0520 P=P+1

M0525 R=1

M0530 IF W-P NZ, GO TO 12

M0535 ZZ =0
S0540 T+P

M0555 Z =Z +1
S0560 0 0

M0565 T=T+P+1

M0575 GO TO 9

M0580 L=M INTEGER(L/M)+M+1

M0585 GO TO 4

15 ROW STOP

RETURN TO SEARCH

```


APPENDIX B-2
DIRECT PATH ROUTINE

18 BEGIN PK PROGRAM

M0590 T=1

M0595 S=1

M0600 R=1

M0605 W=1

M0610 P=0

M0615 B=0

M0620 V=0

M0625 N=-1

M0630 M=K+2

M0635 Y =0
S0640 0

M0645 Y =0
S0650 1

M0655 FF=CC

M0660 XX =CC
S0665 0

BEGIN SEARCH ROUTINE

FIRST XX ARRAY NO. IS SYSTEM INPUT

APPENDIX B-2
DIRECT PATH ROUTINE

M0670	L=0	30	SEARCH SUBSCRIPT
M0675	L=L+1	31	NEW SEZRCH SUBSCRIPT
M0680	IF G ZERO,GO TO 40		TO ROW STOP
S0685	L		
M0690	IF G ZERO,GO TO 41	32	TO COLUMN STOP
S0695	L		
M0700	IF FF-G ZERO,GO TO 34		TEST(+)
S0705	L		
M0710	IF FF+G ZERO,GO TO 33		TEST(-)
S0715	L		
M0720	GO TO 31		
M0725	R=-1	33	NOTE NEGATIVE VARIABLE
M0730	L=M INTEGER(L/M)	34	ASSOCIATED OUTPUT
M0735	IF G -DD ZERO, GO TO 43		COMPLETE PATH TEST
S0740	L		
M0745	IF XX -G ZERO,GO TO 39	35	DUPLICATE POINT TEST
S0750	V+P L		



APPENDIX B-2
DIRECT PATH ROUTINE

M0755 IF XX +G NZ, GO TO 36
S0760 V+P L

M0765 R=1

M0770 GO TO 39

M0775 P=P+1

36

M0780 IF W-P NZ,GO TO 35

M0785 P=0

END OF TEST

M0790 N=N+W+1

37 RECORDING ROUTINE FOR XX ARRAY

M0795 XX =RXX

38 RECORD LINE OF XX ARRAY

S0800 N+P V+P

M0805 R=1

M0810 P=P+1

M0815 IF W-P NZ,GO TO 38

M0820 XX =G

RECORD LAST LINE PT OF XX ARAY

S0825 N+P L

M0830 L=L+M+1

39 INCREASE SEARCH SUBSCRIPT TO NEXTROW

APPENDIX B-2
DIRECT PATH ROUTINE

M0835	P=0		
M0840	R=1		
M0845	GO TO 32		
M0850	L=M INTEGER(L/M)+M+1	40	ROW STOP
M0855	GO TO 32		
M0860	V=V-W	41	COLUMN STOP
M0865	IF V-Y NEG, GO TO 42		TEST V
S0870	W-1		
M0875	FF=XX		NEW SEARCH NO.
S0880	V+W-1		
M0885	GO TO 30		
M0890	IF N-Y NEG, GO TO 45	42	FINISHED
S0895	W		
M0900	V=N		INC ROW SIZE ROUTINE
M0905	FF=XX		NEW SEARCH NO.
S0910	V+W		



APPENDIX B-2 DIRECT PATH ROUTINE

M0915 W=W+1

M0920 Y =N
S0925 W

V-TEST STOP

M0930 N=N-1

M0935 GO TO 30

SEARCH FOR NEW NO.

M0940 YY =XX
S0945 T+P V+P

43

PATH RECORD ROUTINE

M0960 P=P+1

M0965 IF W-P NZ,GO TO 43

M0970 YY =RG
S0975 T+P L

OUTPUT NO.

M0990 P=P+1

M0995 YY =0
S1000 T+P

SPACER FOR END OF PATH

M1015 R=1

M1020 B=B+1

NUMBER OF PATHS TALLY

APPENDIX B-2
DIRECT PATH ROUTINE

RESET FOR NEXT PATH

M1025 T=T+P+1

M1030 P=0

M1035 GO TO 39

SEARCH



APPENDIX B-3
LOOP COMBINATION ROUTINE

M1040 Y =B	45	NUMBER OF PATHS
S1045 0		
M1050 E=0		BEGIN PRODUCT OF LR ROUTINE
M1055 Q=1		PAIRS
M1060 C=1		
M1065 N=0		
M1070 A=2		LOOP NO.
M1075 G =1		FIRST ZZ NUMBER
S1080 1		
M1085 XX =0		
S1090 0		
M1110 IF Z -1 NORZ, GO TO 59		MAX OF ONE LOOP NO PAIRS
S1115 0		
M1120 IF A-Z PNZ,GO TO 58	50	HAVE CHECKED ALL LOOPS
S1125 0		
M1130 IF C-1 PNZ,GO TO 53		IF SO CAN USE Y NO.
M1135 N=N+1	51	



APPENDIX B-3
LOOP COMBINATION ROUTINE

M1140 IF ZZ	NZ,GO TO 51	FIND END OF LOOP
S1145	N	
M1150 IF A-Z	PNZ,GO TO 58	
S1155	0	52
M1160 G =N+1		FIRST ZZ IN LOOP A
S1165	A	
M1170 A=A+1		NEW LOOP NO.
M1175 N=N+1		NEW CHECK NO.
M1180 GO TO 56		
M1185 N=G		
S1190	A	53 START NEXT LOOP
M1195 A=A+1		
M1200 Q=G		ESTABLISH CHECK PT.
S1205	C	
M1210 GO TO 56		
M1215 Q=Q+1		54 SET NEXT CHECK POINT
M1220 IF ZZ	ZERO,GO TO 57	END OF CHECK LOOP
S1225	Q	



APPENDIX B-3
LOOP COMBINATION ROUTINE

```

M1230 N=G
S1235 A-1

M1240 IF ABS(ZZ )-ABS(ZZ )ZERO
S1245 Q N

M1250 , GO TO 50

M1255 N=N+1

M1260 IF ZZ NZ,GO TO 56
S1265 N

M1270 GO TO 54

M1275 XX =C
S1280 E

M1285 XX =A- 1
S1290 E+1

M1305 E=E+2

M1310 Q=G
S1315 C

M1320 IF C-1 ZERO, GO TO 52
55 CHECK NUMBER FOR START NEW LOOP
56 IF ZERO GO TO NEXT LOOP
57 RECORD PAIRS

```


APPENDIX B-3
LOOP COMBINATION ROUTINE

M1325	GO TO 50		
M1330	C=C+1	58	
M1335	IF C-Z ZERO,GO TO 59		COMPLETED THIS PART
S1340	0		
M1345	Q=G		
S1350	C		
M1355	A=C+2		
M1360	GO TO 55		
M1365	Z =E/2		
S1370	1	59	NUMBER OF PAIRS
M1375	IF Z -1 NORZ, GO TO 70		
S1380	1		
M1410	A=0		
M1415	B=A+2	61	
M1420	C=A+1		
M1425	D=B+2	62	



APPENDIX B-3
LOOP COMBINATION ROUTINE

		END OF RECORDED PAIRS
M1430 IF D-2Z ZERO,GO TO 70		
S1435 1		
M1440 IF XX -XX NZ,GO TO 66		
S1445 A B		
M1450 IF XX -XX ZERO,GO TO 67		63
S1455 C D		
M1460 IF XX -XX NEG,GO TO 65		
S1465 C D		
M1470 D=D+2		64
M1475 IF D-2Z NZ,GO TO 63		
S1480 1		
M1485 B=B+2		65
M1490 IF B-2Z NZ,GO TO 62		
S1495 1		
M1500 A=A+2		66
M1505 GO TO 61		
M1510 IF XX -XX NZ,GO TO 64		67
S1515 B+1 D+1		

APPENDIX B-3 LOOP COMBINATION ROUTINE

M1520	XX =XX		
S1525	E	A	
M1530	XX =XX		
S1535	E+1	C	
M1540	XX =XX		
S1545	E+2	B+1	
M1550	E=E+3		
M1565	GO TO 65		
M1570	Z =(E-2Z)/3		70 NUMBER OF PRODUCTS 3 AT A TIME
S1575	2	1	
M1590	A=0		FIRST CHECK NUMBER
M1595	G=2Z +3Z		
S1600	1	2	
M1602	IF Z -1 NORZ,	GO TO 80	CAN HAVE NO PRODUCTS OF 4
S1602	2		
M1605	B=A+2		71 SECOND CHECK NUMBER
M1610	C=A+1		SECOND CHECK POINT

APPENDIX B-3
LOOP COMBINATION ROUTINE

M1615	D=B+2	72	THIRD CHECK NUMBER
M1620	IF D-2Z	73	
S1625	1		
M1630	F=2Z		SEARCH FOR SECOND CHECK POINT
S1635	1		
M1640	IF XX -XX		CHECK NUMBERS ALL RIGHT
S1645	A D		
M1650	IF D-B-2 NZ, GO TO 75		
M1655	A=A+2	74	NEW FIRST CHECK NUMBER
M1660	GO TO 71		
M1665	B=B+2	75	NEW SECOND CHECK NUMBER
M1670	GO TO 72		
M1675	IF XX -XX	76	TWO CHECK POINTS ALL RIGHT
S1680	C F		
M1685	IF XX -XX NEG, GO TO 78		
S1690	C F		
M1695	F=F+3	77	

APPENDIX B-3
LOOP COMBINATION ROUTINE

M1700	IF F-G NZ, GO TO 76		
M1705	D=D+2	78	NEW CHECK NUMBER
M1710	GO TO 73		
M1715	IF XX -XX NZ, GO TO 77	79	THREE CHECK POINTS
S1720	B+1 F+1		
M1725	IF XX -XX NZ, GO TO 77		FOUR CHECK POINTS
S1730	D+1 F+2		
M1735	XX =XX		RECORD PRODUCTS OF FOUR
S1740	E A		
M1745	XX =XX		
S1750	E+1 C		
M1755	XX =XX		
S1760	E+2 B+1		
M1765	XX =XX		
S1770	E+3 D+1		
M1775	E=E+4		STORAGE TALLY
M1790	GO TO 78		

APPENDIX B-3
LOOP COMBINATION ROUTINE

M1795 Z = (E-G) / 4
S1800 3

80 N09 OF PRODUCTS OF FOUR

APPENDIX B-4
LOOPS-NOT-TOUCHING-PATHS ROUTINE

M1805	D=1	85	BEGIN LRK ROUTINE
M1810	K=0		
M1815	F=E	90	
M1820	K=K+1		
M1825	A=1		LOOP NO.
M1830	Q=K	91	PATH CHECK NO.
M1835	IF A-Z PNZ, GO TO 95		END OF TEST
S1840	₀		
M1845	N=G	92	LOOP CHECK NO.
S1850	_A		
M1855	IF ABS(YY)-ABS(ZZ) ZERO	93	TEST
S1860	_Q _N		
M1865	, GO TO 94		
M1870	N=N+1		INC LOOP CHECK NO.
M1875	IF ZZ NZ, GO TO 93		END OF LOOP
S1880	_N		

APPENDIX B-4
 LOOPS-NOT-TOUCHING-PATHS ROUTINE

M1885	Q=Q+1		NEW PATH CHECK NO.
M1890	IF YY	NZ, GO TO 92	
S1895	Q		END OF PATH
M1900	XX =A		
S1905	E		RECORD LOOP NO.
M1910	E=E+1		
M1915	A=A+1		94 NEW LOOP NO.
M1920	GO TO 91		
M1925	Y =E-F		
S1930	D		95 RECORD NO. OF LOOPS
M1935	IF Y -1	NORZ, GO TO 119	
S1940	D		CAN BE NO PAIRS
M1945	D=D+1		START PRODUCTS OF LRK ROUTINE
M1950	A=0		FIRST CHECK PAIR
M1955	B=E-1		LAST XX NO. OF LOOPS
M1960	Q=F		FIRST TEST NO.

APPENDIX B-4
 LOOPS-NOT-TOUCHING-PATHS ROUTINE

M1965	IF XX -XX	ZERO, GO TO 99	96	TEST
S1970	Q	A		
M1975	IF XX -XX	NEG, GO TO 98		
S1980	Q	A		
M1985	A=A+2		97	NEXT CHECK PAIR
M1990	IF A-2Z NZ,	GO TO 96		CHECK FOR END OF PAIR LIST
S1995	1			
M2000	GO TO 102			
M2005	Q=Q+1		98	NEXT TEST NO.
M2010	IF Q-B NZ,	GO TO 96		CHECK END OF TEST NO. LIST
M2015	GO TO 102			END
M2020	C=Q+1		99	SECOND TEST PT.
M2025	IF XX -XX	ZERO, GO TO 101	100	SECOND TEST
S2030	C	A+1		
M2035	IF XX -XX	PNZ, GO TO 97		
S2040	C	A+1		
M2045	C=C+1			NEW SECOND TEST PT.

APPENDIX B-4
 LOOPS-NOT-TOUCHING-PATHS ROUTINE

M2050	IF C-B NORZ, GO TO 100	CHECK END OF LOOP LIST
M2055	GO TO 97	
M2060	XX =XX	
S2065	E Q	101 RECORD PAIRS
M2070	XX =XX	
S2075	E+1 C	
M2090	E=E+2	
M2095	GO TO 97	
M2100	Y =(E-B-1)/2	102 NO. OF PAIRS FOR THIS PATH
S2105	D	
M2110	IF Y -1 NORZ, GO TO 120	CAN BE NO TRIPLES
S2115	D	
M2120	D=D+1	BEGIN TRIPLE ROUTINE
M2125	Q=F	SET FIRST TEST NO.
M2130	A=2Z	SET FIRST CHECK NO.
S2135	1	
M2140	G=A+3Z	CHECK NO. STOP
S2145	2	

APPENDIX B-4
LOOPS-NOT-TOUCHING-PATHS ROUTINE

M2150	L=E		
M2155	IF XX -XX	ZERO, GO TO 106	103
S2160	Q A		FIRST TEST
M2165	IF XX -XX	NEG, GO TO 105	
S2170	Q A		
M2175	A=A+3		104
			NEW FIRST CHECK NO.
M2180	IF A-G NZ,	GO TO 103	
			CHECK FOR END OF CHECK LIST
M2185	GO TO 118		
M2190	Q=Q+1		105
			NEW FIRST TEST NO.
M2195	IF Q-B+1 NZ,	GO TO 103	
			CHECK END OF TEST NO. LIST
M2200	GO TO 118		
M2205	C=Q+1		106
			SET SECOND TEST NO.
M2210	IF XX -XX	ZERO, GO TO 115	107
S2215	C A+1		SECOND TEST
M2220	IF XX -XX	PNZ, GO TO 104	
S2225	C A+1		

APPENDIX B-4
 LOOPS-NOT-TOUCHING-PATHS ROUTINE

M2230	C=C+1		
M2235	IF C-B NZ, GO TO 107		
M2240	GO TO 104		
M2245	H=C+1	115	SET THIRD TEST NO.
M2250	IF XX -XX		
S2255	H A+2		
M2260	IF XX -XX		
S2265	H A+2		
M2270	H=H+1		
M2275	IF H-B NORZ, GO TO 116		
M2280	GO TO 104		
M2285	XX =XX		
S2290	E Q		
M2295	XX =XX		
S2300	E+1 C		
M2305	XX =XX		
S2310	E+2 H		
		117	RECORD TRIPLES



APPENDIX B-4
 LOOPS-NOT-TOUCHING-PATHS ROUTINE

M2325 E=E+3		
M2330 GO TO 104		
M2335 Y=(E-L)/3	118	NO. OF TRIPLES
S2340 D		
M2345 D=D+1		
M2350 GO TO 121		
M2355 D=D+1	119	
M2360 Y=0		NO PAIRS
S2365 D		
M2370 D=D+1	120	
M2375 Y=0		NO TRIPLES
S2380 D		
M2382 D=D+1		
M2385 K=K+1	121	
M2390 IF YY NZ, GO TO 121		FIND FIRST NO. NEXT PATH
S2395 K		
M2400 IF (D-1)/3-Y NZ, GO TO 90		CHECK FOR END OF PATH LIST
S2405 0		

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M2500 P=0 108 BEGIN PERFORMANCE FUNCTION PUNCH

M2501 IF SS ZERO, GO TO 200

M2502 PUNCH MSG, SP4

M2503 BEGIN PERFORMANCE FUNCTION

M2505 S=Z +1
S2510 0

M2511 PUNCH MSG, SP2

M2512 NUMERATOR

M2515 G =1
S2520 S

M2525 T=0

M2530 B=2Z +3Z +4Z
S2535 1 2 3

M2540 C=B

M2545 A=0

M2550 A=A+1

130

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M2555 IF A-Y PNZ, GO TO 155		COMPLETE NUMERATOR
S2560 0		
M2565 Q=G		
S2570 S		
M2575 R=0		
M2580 S=S+1		
M2585 IF YY NEG, GO TO 139		PUNCH SIGN
S2590 Q		
M2595 IF A-1 NZ, GO TO 140		PUNCH SIGN UNLESS FIRST TERM +
M2600 IF ABS(YY)-QQ PNZ, GO TO 132	131	DO NOT PUNCH SUMMER
S2605 Q		
M2610 PUNCH YY		
S2615 Q		
M2620 Q=Q+1	132	
M2625 IF YY NZ, GO TO 131		
S2630 Q		
M2635 G =Q+1		FIRST YY NO. NEXT PATH
S2640 S		

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M2645	IF R ZERO, GO TO 135	PUNCH PATH ONLY
M2650	IF ABS(ZZ)-QQ PNZ, GO TO 134	DO NOT PUNCH SUMMERS
S2655	K	
M2660	PUNCH ZZ	LR FOR THIS PATH
S2665	K	
M2670	K=K+1	134
M2675	IF ZZ NZ, GO TO 133	
S2680	K	
M2685	IF R-1 ZERO, GO TO 141	
M2690	P=P+1	
M2695	IF R-2 ZERO, GO TO 143	
M2700	IF R-3 ZERO, GO TO 149	
M2705	T=T+1	135
M2710	IF Y ZERO, GO TO 151	JUMP TO NEXT PATH
S2715	T	
M2720	D=XX	136 LOOP NO.
S2725	C	

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M2730	N=G			FIRST ZZ NUMBER THIS LOOP
S2735	D			
M2740	IF R ZERO, GO TO 137			NUMBER OF LOOPS LESS 1, THIS TERM
M2745	IF R-1 ZERO, GO TO 138			
M2750	IF R-2 ZERO, GO TO 152			
M2755	IF R-3 ZERO, GO TO 148			
M2760	R=1	137		SET R FOR PAIRS
M2765	K=N	138		SET ZZ NUMBER FOR LOCATING LOOP
M2770	Q=G			SET YY NUMBER FOR LOCATING PATH
S2775	S-1			
M2780	IF YY ZZ NEG, GO TO 140			DETERMINE SIGN
S2785	Q N			
M2790	PUNCH MSG	139		SIGN OF NEXT TERM
M2791	MINUS			
M2795	GO TO 131			
M2800	PUNCH MSG	140		

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M2801	PLUS	
M2805	GO TO 131	
M2810	C=C+1	141 NEXT LR NO.
M2815	IF C-Y -B NZ, GO TO 136	
S2820	T	
M2825	R=2	SET NO. FOR PAIRS
M2830	T=T+1	
M2835	IF Y ZERO, GO TO 151	
S2840	T	
M2845	F=XX	142 FIND SECOND LOOP NO.
S2850	C+1	
M2855	M=G	FIRST ZZ NO. OF LOOP
S2860	F	
M2865	GO TO 136	
M2870	K=N	152
M2872	Q=G	
S2873	S-1	

APPENDIX B-5

DETERMINE SIGN OF NEXT TRIPLE

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M2955 H=XX		
S2960	C+2	
M2965 L=G		
S2970	H	
M2975 GO TO 142		
M2980 K=N		148
M2985 Q=G		
S2990	S-1	
M2995 IF YY ZZ ZZ ZZ NEG, GO TO 140		
S3000	Q N M L	
M3005 GO TO 139		
M3010 IF P-1 ZERO, GO TO 144		149
M3015 IF P-2 NZ, GO TO 150		
M3020 K=L		
M3025 GO TO 145		
M3030 P=0		150

147 FIND NO. OF 3RD LOOP OF TRIPLE
FIRST ZZ NO. THIS LOOP

DETERMINE TERM SIGN

SET FOR RECORDING LOOP

APPENDIX B-5 ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M3035	C=C+3	
M3040	IF C-3Y -2Y -Y -B NZ	
S3045	T T-1 T-2	
M3050	, GO TO 147	
M3055	T=3A	151
M3060	B=C	
M3065	GO TO 130	
M3070	A=0	155
M3075	Q=0	START DENOM ROUTINE
M3080	R=0	
M3081	PUNCH 0, SP4	
M3082	PUNCH MSG, SP2	
M3083	DENOMINATOR	
M3085	PUNCH 1	
M3090	Q=Q+1	160

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M3095	A=A+1		
M3100	IF A-Z	PNZ, GO TO 164	
S3105	0		FINISHED LR, START PAIRS
M3110	IF ZZ	NEG, GO TO 161	
S3115	Q		
M3120	PUNCH MSG		
M3121	MINUS		
M3125	GO TO 162		
M3130	PUNCH MSG	161	
M3131	PLUS		
M3135	IF ABS(ZZ)	-QQ PNZ, GO TO 163	
S3140	Q	162	DO NOT PUNCH SUMMER
M3145	PUNCH ZZ		
S3150	Q		
M3155	Q=Q+1	163	
M3160	IF ZZ	NZ, GO TO 162	
S3165	Q		

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M3170	IF R ZERO, GO TO 160	TO NEXT LR
M3175	GO TO 174	TO NEXT LOOP OF PAIR, ETC.
M3180	P=0	164
M3185	R=1	
M3190	B=0	
M3195	IF Z ZERO, GO TO 185	165 FINISHED
S3200	R	
M3205	C=XX	166 SET UP FOR PAIRS OR FIRST TWO
S3210	B	NUMBERS OF TRIPLES, ETC.
M3215	D=XX	
S3220	B+1	
M3225	N=G	
S3230	C	
M3235	M=G	
S3240	D	
M3245	IF R-1 ZERO, GO TO 171	PAIRS
M3250	IF R-2 ZERO, GO TO 172	TRIPLES

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M3255	IF R-3 ZERO, GO TO 173		
M3260	GO TO 185		
M3265	R=2	167	TRIPLES
M3270	P=0		
M3275	F=XX		
S3280	B+2	168	3RD NO. OF TRIPLE OR FOURS
M3285	L=G		
S3290	F		FIRST ZZ NUMBER LOOP F
M3295	GO TO 165		
M3300	R=3		
M3305	P=0	169	FOURS
M3310	H=XX		
S3315	B+3	170	FOURTH LOOP NO.
M3320	K=G		
S3325	H		
M3330	GO TO 168		

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M3335	IF ZZ ZZ	NEG, GO TO 181	171	DETERMINING SIGN OF PAIR
S3340	N M			
M3345	GO TO 180			
M3350	IF ZZ ZZ ZZ	NEG, GO TO 180	172	SIGN OF TRIPLE
S3355	N M L			
M3360	GO TO 181			
M3365	IF ZZ ZZ ZZ ZZ	NEG, GO TO 181	173	SIGN OF FOURS
S3370	N M L K			
M3375	GO TO 180			
M3380	IF P-1	ZERO, GO TO 177	174	
M3385	IF P-2	ZERO, GO TO 178		
M3390	IF P-3	ZERO, GO TO 179		
M3395	IF R-1	ZERO, GO TO 182		
M3400	IF R-2	ZERO, GO TO 183		
M3405	IF R-3	ZERO, GO TO 184		
M3410	GO TO 185			

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M3415 Q=N	175
M3420 P=P+1	176
M3425 GO TO 162	
M3430 Q=M	177
M3435 GO TO 176	
M3440 IF R-1 ZERO, GO TO 182	178
M3445 Q=L	
M3450 GO TO 176	
M3455 IF R-2 ZERO, GO TO 183	179
M3460 Q=K	
M3465 GO TO 176	
M3470 PUNCH MSG	180
M3471 PLUS	
M3475 GO TO 175	

APPENDIX B-5 ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M3480 PUNCH MSG 181

M3481 MINUS

M3485 GO TO 175

M3490 B=B+2

182

M3495 P=0

M3500 IF B-2Z NZ, GO TO 166
S3505 1

M3510 GO TO 167

M3515 B=B+3

183

M3520 P=0

M3525 IF B-2Z -3Z NZ, GO TO 168
S3530 1 2

M3535 GO TO 169

M3540 B=B+4

184

M3545 P=0

APPENDIX B-5
ASSEMBLY OF PERFORMANCE FUNCTION ROUTINE

M3550 IF B-2Z -3Z -4Z NZ, GO TO 170
S3555 1 2 3

M3560 GO TO 200 185

APPENDIX B-6 COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M3565	Z =0		
S3565	0	190	
M3566	E=0		
M3570	READ VV,DW,FW,RR	200	
M3575	Q=QQ		
M3580	R=RR		
M3585	READ G TO G		READ COEFFICIENT ARRAY
S3590	0 R		
M3595	READ G TO G		READ NUMERATOR STOPS
S3600	R+1 R+Q		
M3605	READ G TO G		READ DENOMINATOR STOPS
S3610	R+Q+2 R+2Q+1		
M3615	G =-3		
S3620	R+Q+1		
M3622	IF VV ZERO,VV=DW		
M3625	W=VV		SET FREQUENCY
M3635	PI=3.14159265		

APPENDIX B-6 COMPUTATION OF FREQUENCY RESPONSE ROUTINE

```

M3637 PUNCH HDG, SP2
M3638 MAG PHASE LOG DECI- ANG FREQ
S3638 RATIOANGLE MR BELS FREQ CPS
M3640 N=0 210
M3642 R=RR
M3645 N=N+1 203 INCREASE COMPONENT NUMBER
M3650 H=R+Q+N
M3655 IF N-Q PNZ, GO TO 202 TEST .FOR END OF COMPONENT LIST
M3660 K=G +3 SET INITIAL CONDITION
S3665 H
M3670 L=G SET FINAL CONDITION
S3675 H+1
M3680 DO TO 204 FOR K=K(3)L
E3685
M3690 XX =SQRT((G -G2 W2) +2 2
S3695 E+K K K+2
E3700 2 2
M3705 G W )
S3710 K+1
  
```


APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

```

M3715 K=G.+3
S3720      H                               SET INITIAL CONDITION

M3725 DO TO 205 FOR K=K(3)L

M3730 XX      =ARCSIN(G W/XX )           205 COMPUTE TERM ANGLE
S3735 E+K+1    K+1      E+K

M3740 K=G +3
S3745      H                               SET INITIAL CONDITION

M3750 DO TO 201 FOR K=K(3)L

E3755
M3760 IF G -G2 W NEG,XX      =PI-
S3765      K K+2      E+K+1

M3770 XX
S3775      E+K+1

M3780 A=1

M3785 K=G +3
S3790      H                               SET INITIAL CONDITION

M3795 M=G
S3800      R+N                               SET FINAL CONDITION

```


APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

```

M3805 DO TO 206 FOR K=K(3)M
M3810 A=AXX
S3815 E+K
206 COMPUTE NUMERATOR MAGNITUDE

M3820 B=0
M3825 K=G +3
S3830 H
SET INITIAL CONDITION

M3835 DO TO 207 FOR K=K(3)M
M3840 B=B+XX
S3845 E+K+1
207 COMPUTE NUMERATOR ANGLE

M3850 C=1
M3855 K=G +3
S3860 R+N
SET INITIAL CONDITION

M3865 DO TO 208 FOR K=K(3)L
M3870 C=CXX
S3875 E+K
208 COMPUTE DENOMINATOR MAGNITUDE

M3880 D=0
M3885 K=G +3
S3890 R+N
SET INITIAL CONDITION

```


APPENDIX B-6

202 RECORD FIRST MAGNITUDE SUBSCRIPT

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M3983	P=G			
S3983	J+2			
M3984	GO TO 255			
M4005	K=E+Z	215	SET STORAGE SUBSCRIPT	
S4010	0			
M4015	L=K+Z		STORAGE SUBSCRIPT	
S4020	0			
M4025	C=L+Z +Z +Z +Z		STORAGE SUBSCRIPT	
S4030	0 1 2 3			
M4035	T=1			
M4040	N=1			
M4045	V=-1	220		
M4050	W=0			
M4055	IF ABS(ZZ)-QQ PNZ, GO TO 222	221	SUMMER, DO NOT RECORD	
S4060	N			
M4065	A=ZZ		COMPONENT NUMBER	
S4070	N			

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M4075	V=V G		
S4080	J+A	LOOP MAGNITUDE	
M4085	W=W+G		
S4090	J+Q+A	LOOP PHASE ANGLE	
M4095	GO TO 223		
M4100	IF ZZ NEG,V=-V		222
S4105	N		
M4110	N=N+1		223
M4115	IF ZZ NZ, GO TO 221		
S4120	N		
M4125	N=N+1		
M4130	XX =V	LOOP MAG.	
S4135	E+T		
M4140	XX =W	LOOP PHASE ANGLE	
S4145	K+T		
M4150	XX =V COS(W)	REAL PART	
S4155	L+T		
M4160	XX =V SIN(W)	IMAG. PART	
S4165	C+T		

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

```

M4180 T=T+1
M4185 IF T-Z NORZ, GO TO 220
S4190 0
IF(+) HAVE COMPLETED ALL LOOPS

M4195 F=0
START PAIRS

M4200 IF Z ZERO, GO TO 227
S4205 1
NO PRODUCT TERMS

M4210 A=XX
S4215 F
224 FIRST LOOP OF PAIR

M4220 D=XX
S4225 F+1
SECOND LOOP OF PAIR

M4230 V=XX XX
S4235 E+A E+D
MAG LOOP PAIR

M4240 W=XX +XX
S4245 K+A K+D
PHASE ANGLE LOOP PAIR

M4250 XX =V COS(W)
S4255 L+T
REAL PART

M4260 XX =V SIN(W)
S4265 C+T
IMAG. PART

```


APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M4280	F=F+2		
M4285	T=T+1		
M4290	IF F-2Z	NZ, GO TO 224	IF ZERO HAVE COMPLETED PAIRS
S4295		1	
M4300	IF Z	ZERO, GO TO 227	NO TRIPLES OR FOURS
S4305		2	
M4310	A=XX		225 FIRST LOOP OF TRIPLE
S4315		F	
M4320	B=XX		SECOND
S4325		F+1	
M4330	D=XX		THIRD
S4335		F+2	
M4340	V=XX	XX XX	MAG LOOP TRIPLE PRODUCT
S4345		E+A E+B E+D	
M4350	W=XX	+XX +XX	PHASE ANGLE LOOP TRIPLE PROD
S4355		K+A K+B K+D	
M4360	XX	=V COS(W)	REAL PART
S4365		L+T	

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

	IMAG. PART	
M4370 XX =V SIN(W)		
S4375 C+T		
M4380 F=F+3		
M4385 T=T+1		
M4390 IF F-2Z -3Z NZ, GO TO 225		IF ZERO HAVE COMPLETED TRIPLES
S4395 1 2		
M4400 IF Z ZERO, GO TO 227		NO PRODUCTS OF 4
S4405 3		
M4410 A=XX		226 FIRST LOOP OF 4 LR
S4415 F		
M4420 B=XX		SECOND
S4425 F+1		
M4430 D=XX		THIRD
S4435 F+2		
M4440 H=XX		FOURTH
S4445 F+3		
M4450 V=XX XX XX XX		MAGNITUDE
S4455 E+A E+B E+D E+H		

APPENDIX B-6 COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M4460	W=XX	+XX	+XX	+XX	PHASE ANGLE
S4465	K+A	K+B	K+D	K+H	
M4470	XX	=V	COS(W)		REAL PART
S4475	L+T				
M4480	XX	=V	SIN(W)		IMAG. PART
S4485	C+T				
M4490	F=F+4				
M4495	T=T+1				
M4500	IF F-2Z	-3Z	-4Z	NZ, GO TO 226	IF ZERO HAVE COMPLETED LR PRODUCTS
S4505	1	2	3		
M4510	A=1			227	
M4515	B=1				
M4520	A=A+XX			228	SUM OF REAL PARTS
S4525	L+B				
M4530	B=B+1				
M4535	IF B-T	NZ, GO TO 228			
M4540	M=0				

APPENDIX B-6 COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M4545	B=1	
M4550	M=M+XX	229 SUM OF IMAGINARY PARTS
S4555	C+B	
M4560	B=B+1	
M4565	IF B-T NZ, GO TO 229	
M4570	X=0	ROUTING IDENTIFIER
E4575	$2 \quad 2$	
M4580	P=SQRT(A +M)	230 COMPUTE MAGNITUDE
M4585	IF P ZERO, GO TO 232	
M4590	IF A NEG, GO TO 231	
M4595	A=ARCSIN(M/P)	COMPUTE PHASE ANGLE
M4600	GO TO 233	
M4605	A=PI-ARCSIN(M/P)	231 COMPUTE (+) PHASE ANGLE
M4610	GO TO 233	
M4615	A=0	232

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M4620	IF X NZ, GO TO 234	233 ROUTING INSTRUCTIONS
M4625	DM=P	DENOMINATOR MAGNITUDE
M4630	DA=A	DENOMINATOR PHASE ANGLE
M4635	IF X-1 ZERO, GO TO 247	234 ROUTING INSTRUCTIONS
M4640	IF X-2 ZERO, GO TO 248	
M4645	F=1	START NUM ROUTINE
M4650	N=1	
M4655	D=2Z +3Z +4Z	STORE SUBSCRIPT
S4660	1 2 3	
M4665	B=C+Z	STORE SUBSCRIPT
S4670	0	
M4675	H=B+Y +Y +Y	235 STORAGE SUBSCRIPT
S4680	F F+1 F+2	
M4685	T=1	
M4690	V=1	INITIAL MAG OF PATH
M4695	W=0	INITIAL PHASE ANGLE

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M4700	IF ABS(YY)-QQ PNZ, GO TO 237	236	TEST FOR SUMMER
S4705	N		
M4710	A=YY		
S4715	N		
M4720	V=V G		MAGNITUDE
S4725	J+A		
M4730	W=W+G		PHASE ANGLE
S4735	J+Q+A		
M4740	GO TO 238		
M4745	IF YY NEG, V=-V	237	
S4750	N		
M4755	N=N+1	238	
M4760	IF YY NZ, GO TO 236		
S4765	N		
M4770	N=N+1		
M4775	IF Y ZERO, GO TO 261		NO LRK, NO PRODUCTS
S4780	F		
M4785	A=XX	239	
S4790	D		

APPENDIX B-6 COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M4795	XX =XX		
S4800	B+T L+A	REAL PART LRK	
M4805	XX =XX		
S4810	H+T C+A	IMAG PART LRK	
M4815	D=D+1		
M4820	T=T+1		
M4825	IF T-Y NORZ, GO TO 239		
S4830	F		
M4835	IF Y ZERO, GO TO 260	NO PRODUCTS THIS PATH	
S4840	F+1		
M4845	A=XX		
S4850	D	240	START PAIRS
M4855	S=XX		
S4860	D+1		
M4865	M=XX XX	MAGNITUDE	
S4870	E+A E+S		
M4875	P=XX +XX	PHASE ANGLE	
S4880	K+A K+S		

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M4885	XX	=M COS(P)	REAL PART PAIRS LR
S4890		B+T	
M4895	XX	=M SIN(P)	IMAGINARY PART
S4900		H+T	
M4905		D=D+2	
M4910		T=T+1	
M4915	IF T-Y	-Y NORZ, GO TO 240	
S4920		F F+1	
M4925	IF Y	ZERO, GO TO 260	NO TRIPLES
S4930		F+2	
M4935	A=XX		241
S4940		D	
M4945	S=XX		
S4950		D+1	
M4955	R=XX		
S4960		D+2	
M4965	M=XX	XX XX	MAGNITUDE TRIPLES LR
S4970		E+A E+S E+R	

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

		PHASE ANGLE TRIPLES LR
M4975	P=XX +XX +XX	
S4980	K+A K+S K+R	
M4985	XX =M COS(P)	REAL PART
S4990	B+T	
M4995	XX =M SIN(P)	IMAGINARY PAAT
S5000	H+T	
M5005	D=D+3	
M5010	T=T+1	
M5015	IF T-Y -Y -Y NORZ	
S5020	F F+1 F+2	
M5025	,GO TO 241	
M5030	A=1	260
M5035	S=1	
M5040	A=A+XX	242 SUM OF REAL PARTS
S5045	B+S	
M5050	S=S+1	
M5055	IF S-T NZ, GO TO 242	

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

```

M5060 M=0
M5065 S=1
M5070 M=M+XX
S5075 H+S
243 SUM OF IMAGINARY PARTS

M5080 S=S+1
M5085 IF S-T NZ, GO TO 243
M5090 X=1
ROUTING IDENTIFIER
M5095 GO TO 230
TO COMPUTE MAG AND PA SUM LRK
M5100 V=VP
247 MAGNITUDE
M5105 W=W+A
PHASE ANGLE
M5110 XX =V COS(W)
S5115 B+1
261 REAL PART
M5120 XX =V SIN(W)
S5125 B+2
IMAGINARY PART
M5130 IF F+2-3Y ZERO, GO TO 245
S5135 0
244 COMPLETED ALL PATHS

```


APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M5140	F=F+3		
M5145	B=B+2		
M5150	GO TO 235	NEXT PATH	
M5155	F=1		245
M5156	B=B-2(Y -1)		
S5156	0		
M5160	A=0		
M5165	M=0		
M5170	A=A+XX		246 SUM OF REAL PARTS
S5175	B+F		
M5180	M=M+XX		SUM OF IMAGINARY PARTS
S5185	B+F+1		
M5190	F=F+2		
M5195	IF F-2Y -1 NZ, GO TO 246		
S5200	0		
M5205	X=2		ROUTING IDENTIFIER

APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

```

M5210 GO TO 230
M5212 IF DM ZERO,GO TO 256      248
M5215 M=P/DM
M5220 P=A-DA
M5230 IF P NEG, GO TO 249
M5235 P=(P-2PI INTEGER(P/2PI))180/PI 255 PHASE ANGLE IN DEGREES LESS THAN 360
M5237 IF M ZERO, GO TO 258      250
M5240 AA=LOG(M)/2.30258509
M5241 PUNCHM,P,AA,(20AA),VV,(VV/2PI)
M5245 VV=VV+DW      257
M5247 W=VV
M5250 IF W-FW NORZ, GO TO 210
M5252 EXIT
M5253 PUNCH M,0,0,0,VV,(VV/2PI) 258

```


APPENDIX B-6
COMPUTATION OF FREQUENCY RESPONSE ROUTINE

M5254 GO TO 257
M5255 $P = (P + 2\pi(\text{INTEGER}(\text{ABS}(P)/2\pi)) + 1) \cdot 249$
M5260 $) \cdot 180/\pi$
M5265 GO TO 250
M5270 PUNCH MSG 256
M5275 MAG RATIO INFINITE, PHASE
S5275 ANGLE 180 DEGREES, FREQ IS
M5280 PUNCH VV,SP2
M5285 GO TO 257
M5290 START AT 16

APPENDIX C PERFORMANCE FUNCTION

BEGIN PERFORMANCE FUNCTION

NUMERATOR		
1	60000000	51
MINUS		
2	60000000	51
3	20000000	51
PLUS		
4	60000000	51
5	30000000	51
PLUS		
6	60000000	51
7	40000000	51
PLUS		
8	60000000	51
9	40000000	51
10	50000000	51
MINUS		
11	60000000	51
12	20000000	51
13	30000000	51
MINUS		
14	60000000	51
15	20000000	51
16	40000000	51
MINUS		
17	60000000	51

APPENDIX C PERFORMANCE FUNCTION

18	20000000	51
19	40000000	51
20	50000000	51
PLUS		
21	60000000	51
22	30000000	51
23	40000000	51
PLUS		
24	60000000	51
25	30000000	51
26	40000000	51
27	50000000	51
MINUS		
28	60000000	51
29	20000000	51
30	30000000	51
31	40000000	51
MINUS		
32	60000000	51
33	20000000	51
34	30000000	51
35	40000000	51
36	50000000	51
PLUS		
37	70000000	51
38	30000000	51
39	40000000	51
40	50000000	51
41	60000000	51
MINUS		

APPENDIX C PERFORMANCE FUNCTION

42	70000000	51
43	30000000	51
44	40000000	51
45	50000000	51
46	60000000	51
47	20000000	51
PLUS		
48	10000000	51
49	20000000	51
50	30000000	51
51	40000000	51
52	50000000	51
53	60000000	51
54		

DENOMINATOR

55	10000000	51
PLUS		
56	60000000	51
PLUS		
57	70000000	51
58	30000000	51
59	40000000	51
60	50000000	51
61	60000000	51
PLUS		
62	10000000	51

APPENDIX C PERFORMANCE FUNCTION

63	20000000	51
64	30000000	51
65	40000000	51
66	50000000	51
67	60000000	51
MINUS		
68	20000000	51
PLUS		
69	30000000	51
PLUS		
70	40000000	51
PLUS		
71	40000000	51
72	50000000	51
PLUS		
73	60000000	51
MINUS		
74	60000000	51
75	20000000	51
PLUS		
76	60000000	51
77	30000000	51
PLUS		
78	60000000	51
79	40000000	51
PLUS		
80	60000000	51
81	40000000	51
82	50000000	51
MINUS		

APPENDIX C PERFORMANCE FUNCTION

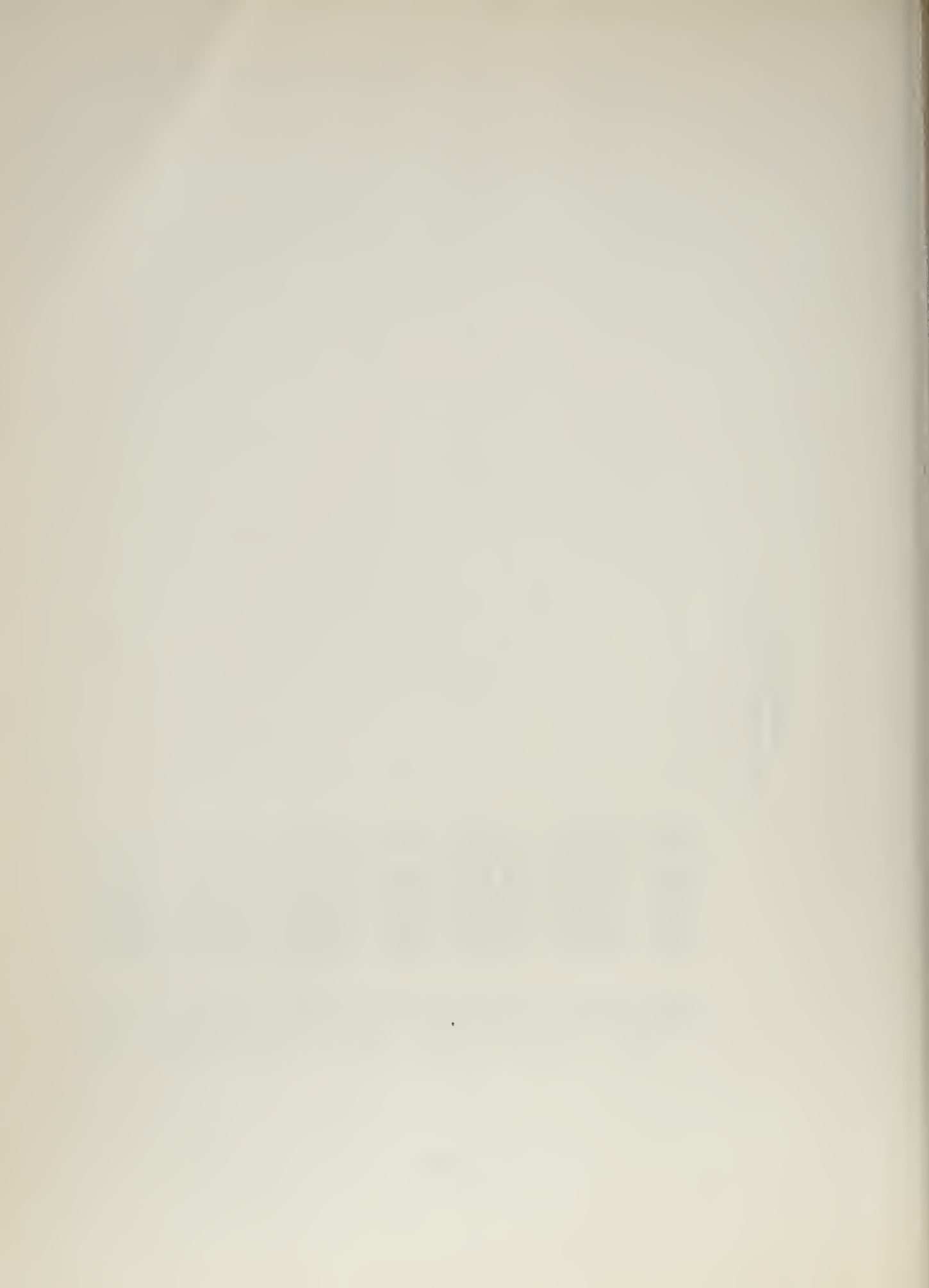
83	70000000	51
84	30000000	51
85	40000000	51
86	50000000	51
87	60000000	51
88	20000000	51
MINUS		
89	20000000	51
90	30000000	51
MINUS		
91	20000000	51
92	40000000	51
MINUS		
93	20000000	51
94	40000000	51
95	50000000	51
MINUS		
96	20000000	51
97	60000000	51
PLUS		
98	30000000	51
99	40000000	51
PLUS		
100	30000000	51
101	40000000	51
102	50000000	51
PLUS		
103	30000000	51
104	60000000	51

APPENDIX C PERFORMANCE FUNCTION

PLUS		
105	40000000	51
106	60000000	51
PLUS		
107	40000000	51
108	50000000	51
109	60000000	51
MINUS		
110	60000000	51
111	20000000	51
112	30000000	51
MINUS		
113	60000000	51
114	20000000	51
115	40000000	51
MINUS		
116	60000000	51
117	20000000	51
118	40000000	51
119	50000000	51
PLUS		
120	60000000	51
121	30000000	51
122	40000000	51
PLUS		
123	60000000	51
124	30000000	51
125	40000000	51
126	50000000	51
MINUS		

APPENDIX C PERFORMANCE FUNCTION

127	20000000	51
128	30000000	51
129	40000000	51
MINUS		
130	20000000	51
131	30000000	51
132	40000000	51
133	50000000	51
MINUS		
134	20000000	51
135	30000000	51
136	60000000	51
MINUS		
137	20000000	51
138	40000000	51
139	60000000	51
MINUS		
140	20000000	51
141	40000000	51
142	50000000	51
143	60000000	51
PLUS		
144	30000000	51
145	40000000	51
146	60000000	51
PLUS		
147	30000000	51
148	40000000	51
149	50000000	51
150	60000000	51



APPENDIX C PERFORMANCE FUNCTION

MINUS			
151	60000000	51	
152	20000000	51	
153	30000000	51	
154	40000000	51	
MINUS			
155	60000000	51	
156	20000000	51	
157	30000000	51	
158	40000000	51	
159	50000000	51	
MINUS			
160	20000000	51	
161	30000000	51	
162	40000000	51	
163	60000000	51	
MINUS			
164	20000000	51	
165	30000000	51	
166	40000000	51	
167	50000000	51	
168	60000000	51	

APPENDIX C

The numbers on the preceding pages of this Appendix, represent the component numbers of each term of the performance function of the sample system of Chapter 4. To illustrate the complexity of the system more clearly, the performance function is written in the more conventional form below. Since there are so many terms the numerator and denominator are written separately.

NUMERATOR:

$$\begin{aligned} &G_6 - G_6 G_2 + G_6 G_3 + G_6 G_4 + G_6 G_4 G_5 - G_6 G_2 G_3 - G_6 G_2 G_4 - \\ &- G_6 G_2 G_4 G_5 + G_6 G_3 G_4 + G_6 G_3 G_4 G_5 - G_6 G_2 G_3 G_4 - G_6 G_2 G_3 G_4 G_5 \\ &+ G_7 G_3 G_4 G_5 G_6 - G_7 G_3 G_4 G_5 G_6 G_2 + G_1 G_2 G_3 G_4 G_5 G_6 \end{aligned}$$

DENOMINATOR:

$$\begin{aligned} &1 + G_6 + G_7 G_3 G_4 G_5 G_6 + G_1 G_2 G_3 G_4 G_5 G_6 - G_2 + G_3 + G_4 + G_4 G_5 + \\ &+ G_6 - G_6 G_2 + G_6 G_3 + G_6 G_4 + G_6 G_4 G_5 - G_7 G_3 G_4 G_5 G_6 G_2 - \\ &- G_2 G_3 - G_2 G_4 - G_2 G_4 G_5 - G_2 G_6 + G_3 G_4 + G_3 G_4 G_5 + G_3 G_6 + \\ &+ G_4 G_6 + G_4 G_5 G_6 - G_6 G_2 G_3 - G_6 G_2 G_4 - G_6 G_2 G_4 G_5 + G_6 G_3 G_4 + \\ &+ G_6 G_3 G_4 G_5 - G_2 G_3 G_4 - G_2 G_3 G_4 G_5 - G_2 G_3 G_6 - G_2 G_4 G_6 - \\ &- G_2 G_4 G_5 G_6 + G_3 G_4 G_6 + G_3 G_4 G_5 G_6 - G_6 G_2 G_3 G_4 - G_6 G_2 G_3 G_4 G_5 \\ &- G_2 G_3 G_4 G_6 - G_2 G_3 G_4 G_5 G_6 \end{aligned}$$



APPENDIX D FREQUENCY RESPONSE

	MAG RATIO	PHASE ANGLE	LOG MR	DECI- BELS	ANG FREQ	FREQ CPS						
169	12900474	51	35884884	53	11060568	50	22121136	51	10000000	50	15915494	49
170	14044315	51	35462525	53	14750057	50	29500114	51	50000000	50	79577470	49
171	17747787	51	35332349	53	24914421	50	49828842	51	90000000	50	14323945	50
172	27549435	51	96593807	51	44011268	50	88022536	51	13000000	51	20690142	50
173	21079358	51	61045065	52	32385739	50	64771478	51	17000000	51	27056340	50
174	93111771	50	62813241	52	-30995410	49	-61990820	50	21000000	51	33422538	50
175	57900228	50	41767849	52	-23731972	50	-47463944	51	25000000	51	39788735	50
176	47055448	50	16725010	52	-32739008	50	-65478016	51	29000000	51	46154933	50
177	40345055	50	35320177	53	-39420968	50	-78841936	51	33000000	51	52521130	50
178	33393728	50	33964546	53	-47633510	50	-95267020	51	37000000	51	58887328	50
179	29723627	50	33346452	53	-52689818	50	-10537964	52	41000000	51	65253526	50
180	27822516	50	32916660	53	-55560357	50	-11112071	52	45000000	51	71619723	50
181	26530173	50	32553599	53	-57625992	50	-11525198	52	49000000	51	77985921	50
182	25471882	50	32235458	53	-59393896	50	-11878779	52	53000000	51	84352119	50
183	24527121	50	31954082	53	-61035342	50	-12207068	52	57000000	51	90718316	50

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

1898

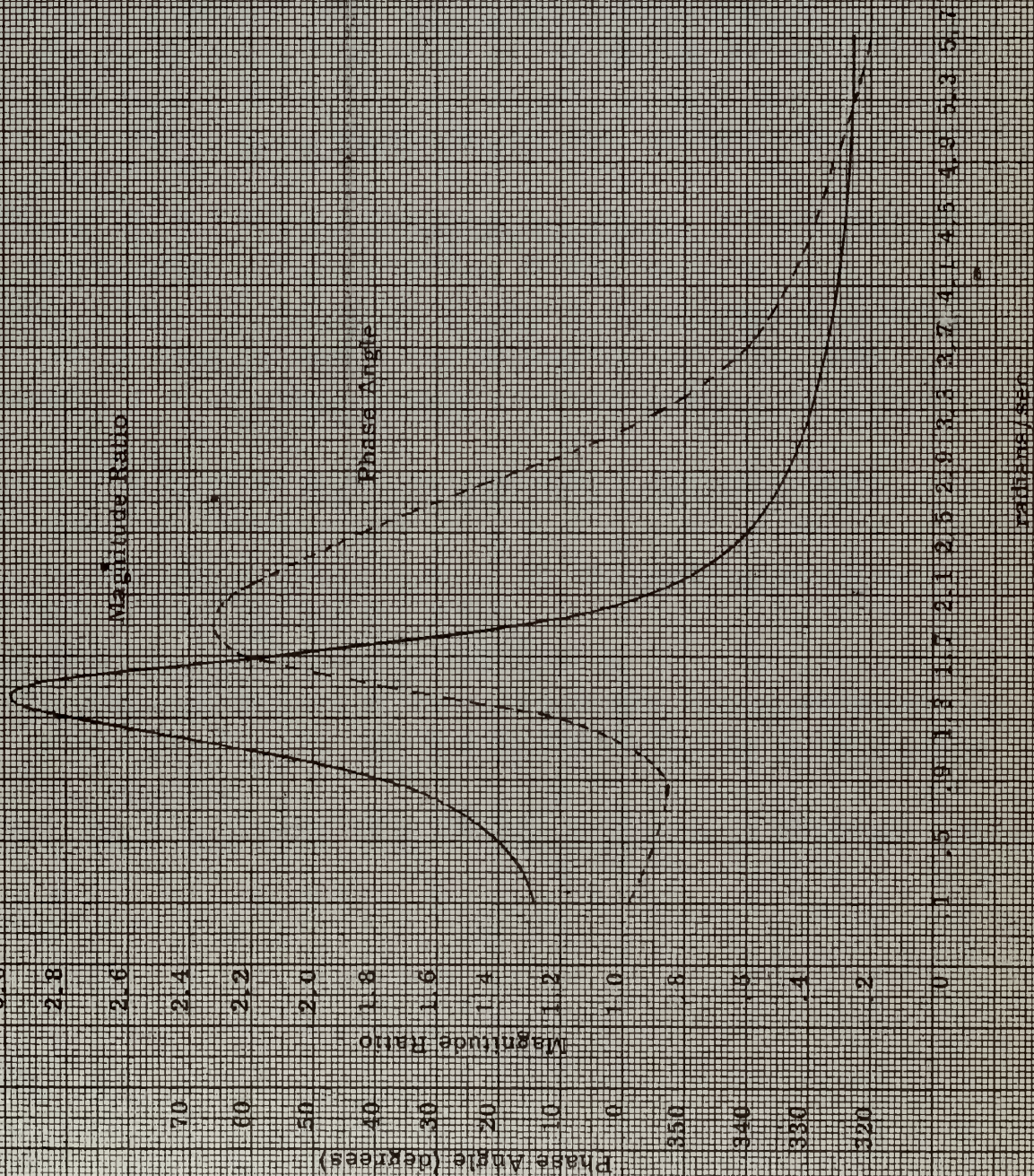
1898

1898

1898

1898

APPENDIX D



BIBLIOGRAPHY

1. Draper, McKay and Lees, Instrument Engineering, Vol. I, II, and III, McGraw-Hill Publications in Aeronautical Science, New York 1955.
2. Thaler, G. J., Servomechanism Analysis, McGraw-Hill, New York, 1953
3. Thaler, G. J., Notes on Compensation Theory, EE 676, U.S. Naval Postgraduate School, Monterey, California, 1959
4. Chu, Yoaham, A Generalized Theory of Linear Multi-Loop Automatic Control Systems, doctoral dissertation, MIT, 1953
5. Mason S. J., Feedback Theory, Research Laboratory of Electronics, MIT, 1955
6. MAC Translator Manual, Instrumentation Laboratory, MIT, Cambridge, Mass., 1959
7. Computing Devices Notes, No. 11, Instrumentation Laboratory, MIT, Cambridge, Mass., 1959
8. IBM 650 Data Processing System Bulletins
 - a. General Information, Console Operation, Special Devices
 - b. 533 Card Read Punch, 537 Card Read Punch, 407 Accounting Machines
 - c. Basic Operation Codes, Program Optimizing, Program Loading

thesS574

A simplified method for digital computat



3 2768 002 00886 4

DUDLEY KNOX LIBRARY